2. Sonderausgabe

DOS EXTRA

18,– DM 145 Ös 18,– sfr

Sonderausgabe DOS-EXTRA

Nr.2 1988

DMV-Verlag

Eine Publikation der
DOS-INTERNATIONAL

3D-CAD-Programm

Komfortable Benutzeroberfläche DOS-Commander

DOS-Utilities Hilfsprogramme für jeden Zweck

SForth 3.1 Ein Compiler im Selbstbau

Alle Listings mit automatischer Fehlerkorrektur

SchneiderData

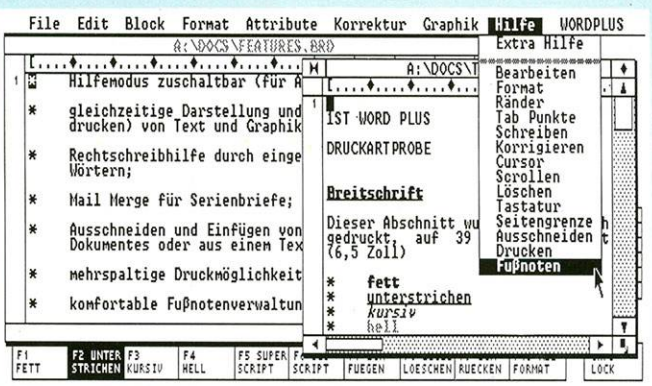
Schreiben Sie erster Klasse mit ...

1st WORD PLUS

Textverarbeitungssystem der Spitzenklasse für IBM-PC, Schneider-PC und Kompatible:

- ★ *außergewöhnlich einfach zu erlernen und zu bedienen*
- ★ *vollständig GEM™-gesteuert, mit Mausunterstützung, Drop-Down-Menüs, und bis zu vier Textfenstern gleichzeitig*
- ★ *ständiger Hilfe-Modus zuschaltbar (für Anfänger)*
- ★ *Ausschneiden und Einfügen von Blöcken innerhalb eines Dokumentes, oder aus einem Textfenster in ein anderes*
- ★ *gleichzeitige Darstellung, Bearbeitung und Ausdruck von Text und Graphik*
- ★ *Serienbriefgestaltung (1st Mail)*
- ★ *alle Druckarten genau am Bildschirm dargestellt (WYSIWYG), wie Fettschrift, Unterstreichen, Kursivschrift, Entwurfsdruck, Hoch- und Tiefstellen, Sonderzeichen, usw.*
- ★ *links- und rechtsbündiger und zentrierter Text*
- ★ *komfortable Such- und Ersetzfunktionen*
- ★ *mehrspaltige Druckmöglichkeit (1st Mail)*
- ★ *automatische Fußnotenverwaltung*
- ★ *eingebautes Wörterbuch zur Rechtschreibhilfe mit 40000 Wörtern, einschließlich Funktionen zum Durchblättern, Hinzufügen und Löschen von Wörtern*

Text und Graphik gleichzeitig darstellen und ausdrucken

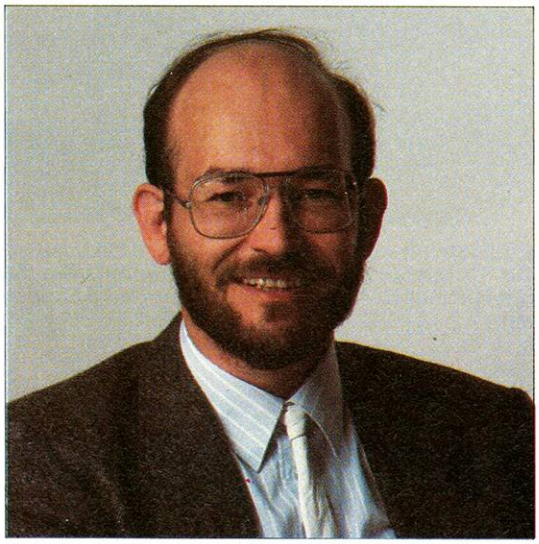


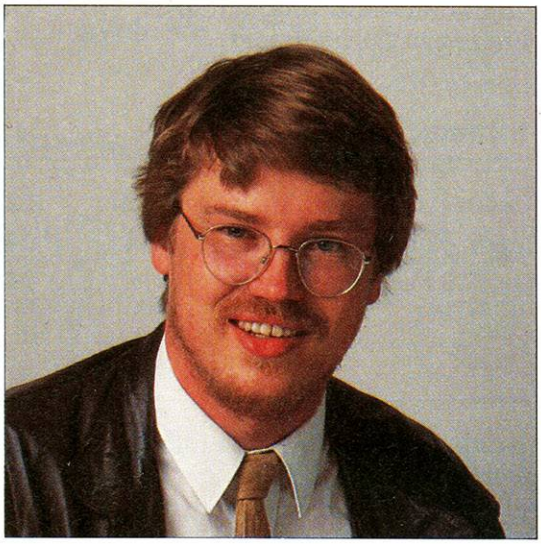
Einfachste Bedienung durch die komfortable GEM-Steuerung

1st WORD PLUS *die Textverarbeitung, die keine Wünsche offen läßt* **DM 299,--**

SchneiderData · Rindermarkt 8 · D-8050 Freising · Telefon 08161/2877

Lehrt die Großen Angst und Schrecken





Software wird immer billiger. Doch die Qualität der Programme steigt. Ein Widerspruch? Aber nein!

Spätestens seitdem die Preise für Personalcomputer drastisch gesunken sind, haben sich auch die Preise für Softwarepakete kontinuierlich nach unten entwickelt. Sogar Firmen, die normalerweise eine Hochpreispolitik fahren, haben abgemagerte Versionen Ihrer Highlights zu Preisen von nur wenigen hundert Mark auf den Markt gebracht. Dabei handelt es sich um Software, die bei der Einführung der heute als »abgespeckt« geltenden Programme den Ruf der Produkte begründete.

Warum hat es solch einen Rutsch nach unten gegeben? Zum einen führte die große Zahl verkaufter PCs natürlich zu höherer Nachfrage, und höhere Verkaufszahlen führen bekanntlich auch zu niedrigeren Preisen. Doch das allein ist es nicht. Vielmehr haben Newcomer in der Branche den Großen Angst gemacht. Neue Programmierer haben Softwarepakete geschrieben, die teilweise ihren Vorbildern bis aufs Haar gleichen. Und diese Programme kosten sehr wenig Geld!

Inwieweit man die Politik dieser kleinen Häuser befürwortet, muß sich jeder selbst überlegen. Zum einen sorgen sie dafür, daß sich jeder die Software kaufen kann, die er braucht. Zum anderen wird die Angelegenheit aber für die Software-Entwickler, die sich als erste Gedanken um ein neues Produkt machen, zunehmend uninteressanter, wenn kurze Zeit später ein Clone den Rahm abschöpft. Hier treten Probleme auf, die die gesamte Entwicklung von neuen Programmen beeinflußen können.

Doch viele Produkte erfordern auch viele Programmierer. Und diese Vielzahl kann vielleicht die finanziell bedingte, mangelnde Lust mancher Entwicklungsfirmen auffangen.

Hier setzt das Ziel einer Zeitschrift wie DOS International an. Durch Informationen über den Computer und viele Tips und Tricks lernen immer mehr Computerbesitzer ihr Gerät besser kennen und verstehen – und werden so zu den Software-Programmierern von morgen. Aus Programmen, die als Listings in der Zeitschrift abgedruckt sind, können Sie Informationen ziehen, wie Sie mit Ihrem Computer besser arbeiten.

DOS Extra 2 ist ein Listing-Sonderheft. Sie finden fertige Programme für fast jedes Einsatzgebiet Ihres Computers. Ganz klar, daß wir der Tradition der DOS entsprechend wieder den Schwerpunkt auf Utilities und Tips und Tricks gelegt haben. Doch auch eine Benutzeroberfläche – der DOS-Commander – und ein CAD-Programm stürzt Sie jetzt nicht mehr in den finanziellen Ruin. Sie können das Programm abtippen (alle Routinen sind in C, Pascal oder Basic geschrieben) oder fertig auf der Databox kaufen. Und ein Spiel nach einem langem Computertag entspannt und lenkt Sie von den täglichen Problemen ab.

In der zweiten Sonderausgabe der DOS International finden Sie Programmtips und Lösungswege, die Sie später in eigene Software einbauen können, um damit den großen Softwarehäusern durch eigene Programme Konkurrenz zu machen. Der Bereich der Software wird damit wieder ein bißchen vielfältiger, und Sie haben die Chance, sich mit Ihrem Hobby ein Nebeneinkommen zu sichern – oder vielleicht sogar ein neues Softwarehaus aufzubauen...
Das DOS Extra 2 ist damit wie die erste Sonderausgabe der DOS International eine Investition für die Zukunft.

Ihr

Andreas Hagedorn

Volker Everts

Tips zur Eingabe:

Geprüfte Eingabe 6
– *Checksummer*

Grafik:

⇨ Ein CAD-Programm selbst programmiert 10
– *Grafikprogramm in Turbo Pascal*

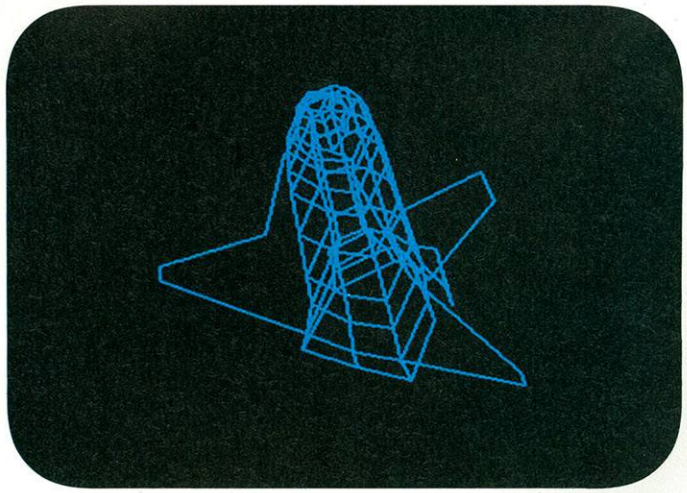
Grafik-Utilities:

Scroll-Routine für beliebige Bildschirmfenster 22
Schnelle Rahmen-Routinen für jeden Zweck 24
Hires-Grafik mit 16 Farben am Schneider PC 1512 26

Utilities:

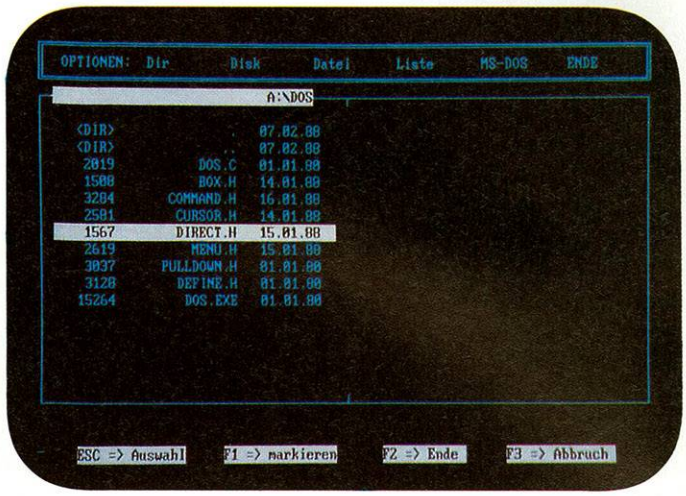
⇨ DOS-Commander macht MS-DOS einfach 28
– *Benutzeroberfläche*
Struktur gefällig? 33
– *Listings automatisch strukturiert*
SCopy, das professionelle Kopierprogramm 38
– *menügesteuertes Kopieren*
Filed – der Editor, dem nichts verborgen bleibt 54
– *nie wieder Probleme mit Steuerzeichen*
So sieht Ihre Datei wirklich aus 66
»Head« schafft Übersicht 69
– *automatische Ausgabe der ersten zehn Zeilen*
Move – Verschieben ohne Bewegung 71

⇨ DOS-Utilities –
nützliche Routinen für jeden Zweck 74
– *Die Include-Datei »Library«*
– *»LibraryE« – Erweiterungen für die Toolbox »Library«* 76
– *Schnelle Fensterverwaltung mit der Window-Toolbox* 77
– *»Tools« enthält Routinen, die Sie nicht so oft brauchen* 78
– *Die Include-Datei »Readstr.inc« dient ausschließlich zur Eingabe von Zeichenketten* 79
– *Datum- und Zeitangabe mit »Readdat.inc«* 81
– *Komfortable Eingabe von Integer-Zahlen* 82
– *Bitweise gepackt mit Turbo Pascal* 83
– *Mehr Übersicht durch Pulldown-Menüs* 84
– *Inhaltsverzeichnis mal ganz anders* 85
– *Tastaturbelegung selbst programmiert* 99
– *Menügesteuerter Programmaufruf mit »Execute«* 105
– *Dateien löschen mit Sicherheitsabfrage* 107
– *Datum- und Zeitänderung von Datei-Einträgen* 113



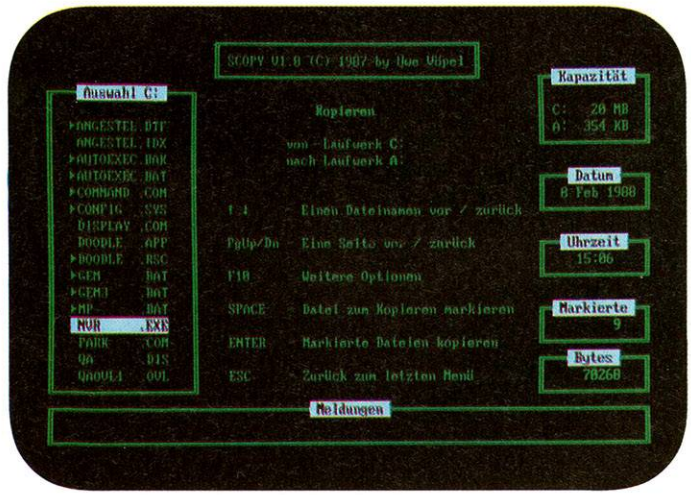
Grafik auf dem Computer ruft immer wieder Bewunderung hervor. CAD heißt das Wort, das alle Computerbesitzer verzaubert. Ein leistungsstarkes CAD-Programm zum Abtippen finden Sie auf

S. 10



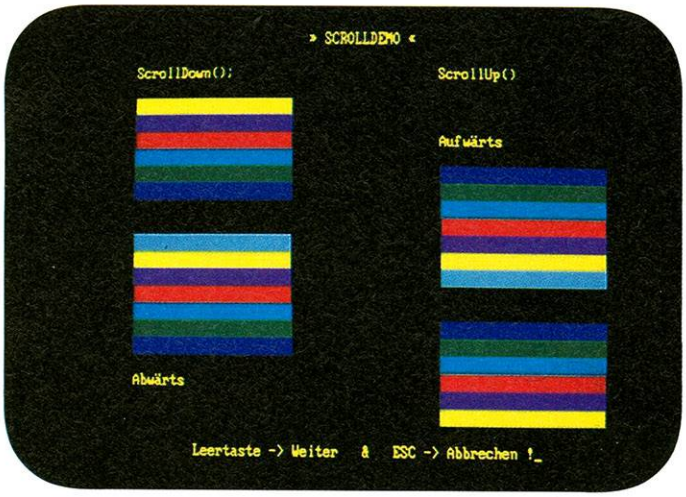
Eine Benutzeroberfläche erleichtert die Arbeit mit dem Betriebssystem MS-DOS. Der DOS-Commander hilft bei fast jedem Befehl – ist aber offen für weitere Routinen.

S. 28



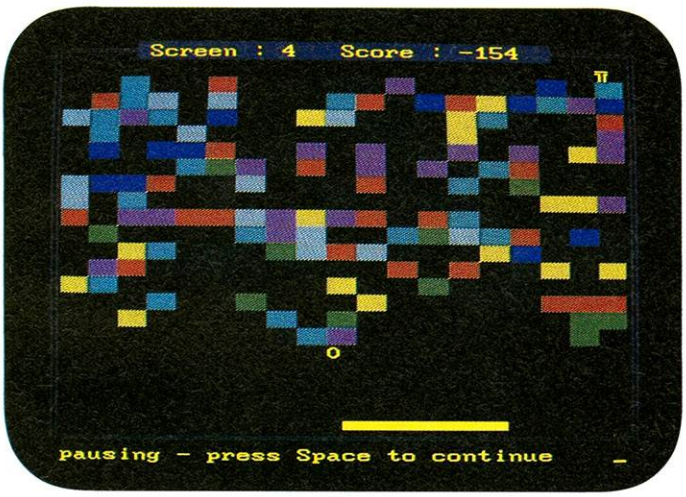
Komfortables Kopieren verschiedenster Dateien in unterschiedlichen Verzeichnissen erlaubt SCopy. Alle Anweisungen werden menügesteuert aufgerufen.

S. 38



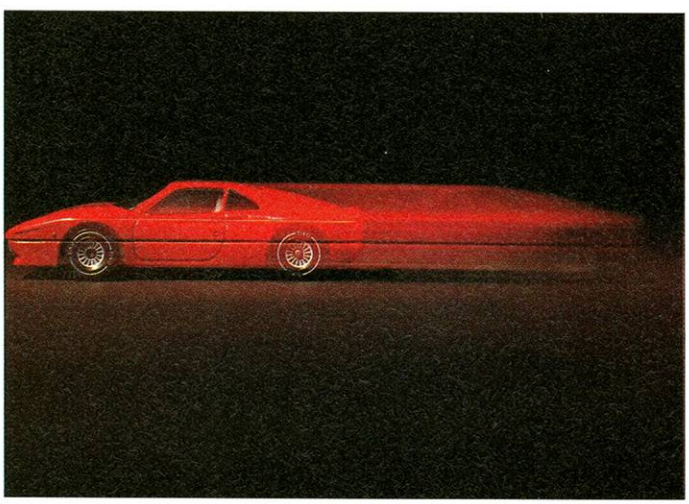
Bewegungen rechts nach oben, links nach unten und gleichzeitig beides auf dem Bildschirm, das geht mit der Routine auf

S. 22



Spielen Sie gerne mit Ihrem Computer? Dann sind Sie bei »Hurrican« richtig. Geschick und Reaktionsfähigkeit ist notwendig, um nicht in der ersten Spielstufe hängenzubleiben.

S. 127



Einen Compiler zum Selbstbauen finden Sie mit SForth. Viele unserer Leser kennen das Projekt schon aus der DOS International. Die neuste Version 3.1 weist aber neue leistungsstarke Befehle auf.

S. 140

Drucker-Utilities:

Fonteditor
– *Zeichensatz für 24-Nadel-Drucker* **116**

Softwareumschalter für Druckerkanäle **123**
– *Umschalten zwischen zwei Druckerports*

Briefumschläge selbst gedruckt **125**
– *individuelle Post*

Spiele:

Hurrican – Schnell wie der Wind **127**
– *Actionspiel mit mehr als zehn Bildern*

Super-Mind, die Master-Mind-Version
für Ihren PC **136**
– *Strategiespiel*

SForth:

⇨ SForth 3.1: Ein Compiler im Selbstbau **140**
– *Neue Version des bekannten DOS-Software-Projekts*

Strings mit SForth **162**

Die neuen Systemworte von SForth 3.1 **166**

Rubriken:

Editorial **3**

Impressum **170**

Sämtliche Routinen und Programme gibt es für 45,– Mark auf 4 Disketten 5 1/4“ oder auf 2 Disketten 3 1/2“

– auch mit Anpassung für die Herkules-Grafikkarte

Geprüfte Eingabe

Mit SCheck, dem Checksummenprüfprogramm der DOS International, werden alle Listings auf korrekte Eingabe überprüft. Der eingebaute Editor erlaubt sofortige Fehlerkorrektur. Sie sparen damit jede Menge Zeit beim Abtippen von Listings.

Immer mehr Leser der DOS International kaufen sich keine fertige Software, sondern finden in der Zeitschrift Programme abgedruckt, die Ihren Wünschen voll entsprechen. Nicht zuletzt Ihre Wünsche, doch einmal ein ganzes Heft voller Programme für den PC zu veröffentlichen, haben zu der zweiten DOS Extra geführt.

Damit nun aber die Eingabe möglichst schnell und fehlerfrei erfolgen kann, finden Sie ab sofort in der DOS sämtliche Listings mit Prüfsumme versehen. Mit Hilfe der Routine SCheck (Listing 1) können Sie Ihre Zeilen mit dem abgedruckten Programm auf Fehler überprüfen. SCheck stellt Ihnen dazu zwei verschiedene Eingabevarianten zur Verfügung.

Beim ersten Weg geben Sie mit dem Editor der entsprechenden Programmiersprache – oder auch mit einem beliebigen Texteditor, der ASCII-Dateien erzeugt – nur den Quellcode ein. Dieser steht hinter der Prüfsumme (in eckigen Klammern) und der Zeilennummer (Zahl mit Doppelpunkt). Listing 2 zeigt eine Pascal-Routine, wie sie im Heft abgedruckt ist, Listing 3 die gleichen Programmzeilen, so wie Sie sie in den Computer eingeben müssen.

Eine Programmzeile wird immer durch die Zeilennummer angezeigt und kann dabei im Druck über mehrere Zeilen gehen. In solch einem Fall müssen Sie die Anweisungen durchgängig in einer Zeile in den Editor eintippen.

Als nächstes starten Sie SCheck mit dem Dateinamen

```
scheck <Name.Ext>
```

Legen Sie sich bitte von der eingegebenen Datei eine Kopie an. Zwar arbeitet SCheck nicht mit dem Original, sondern mit einer Kopie, die erst ganz zum Schluß als aktuelle Datei verändert wird, doch wenn ein Fehler beim Speichern auftritt, so ist es nicht ausgeschlossen, daß die mühsam eingetippten Daten verloren gehen. Also immer als erstes eine Kopie der Daten anlegen.

Das Bild zeigt die aktuelle Zeile in dem oberen Fenster. Die Checksumme wird darunter links angezeigt. Diese vergleichen Sie mit der abgedruckten Zahl in den eckigen Klammern. Rechts daneben steht die Zeilennummer und der Name der aktuellen Datei. Im unteren Fenster werden Fehlermeldungen und Hinweise auf die weitere Bearbeitung ausgegeben. Im hier abgebildeten Fall können Sie mit [Return] die Zeile unverändert übernehmen oder mit [E] den Editor aufrufen.

Der Editor dient der Korrektur fehlerhafter Zeilen. Er arbeitet immer im Einfügemodus, so daß fehlerhafte Eingaben extra gelöscht werden müssen. Sie können also nichts versehentlich überschreiben. Mit den Cursortasten steuern Sie den Cursor im Editierfenster. [Home] bringt den Cursor an den Anfang der aktuellen Programmzeile und [End] ans Ende. Löschen können Sie das Zeichen links vom Cursor mit der Taste oberhalb der Enter-Taste (auf der Schreibtastatur). [Del] löscht das Zeichen genau an der Cursorposition. Mit [Return] übernehmen Sie die geänderte Programmzeile. Die Leerzeichen vor der ersten gültigen Anweisung einer Programmzeile werden übrigens bei der Prüfsummenberechnung nicht berücksichtigt. Strukturen können Sie somit nach eigener Wahl einbauen. Weiter

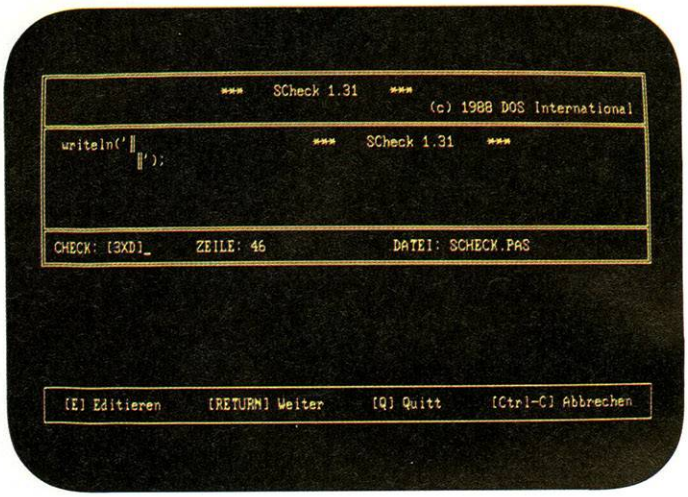


Bild. Mit SCheck können Sie auch falsche Zeilen korrigieren

hinten stehende Leerzeichen sind für die Prüfsummenberechnung allerdings relevant. Die Tabulatorfunktion dürfen Sie nicht verwenden, wenn Sie mit der Prüfsumme korrekt arbeiten wollen.

Haben Sie so das gesamte Programm durchgearbeitet, so speichert ein letzter Druck auf [Enter] das korrigierte Listing. Das Programm wird beendet und das fehlerfreie Programm steht für die weitere Bearbeitung mit einem Compiler, Assembler oder Interpreter zur Verfügung.

Listing 4 zeigt ein Basic-Programm, wie es in der DOS International abgedruckt wird. Die Eingabe erfolgt wiederum ohne Prüfsumme, aber in diesem Fall mit Zeilennummer. Listing 5 zeigt, wie die Routine eingegeben werden muß.

SCheck erlaubt aber noch einen zweiten Weg, das Listing korrekt einzugeben. Diesmal geben Sie die Prüfsumme mit ein. Starten Sie den Programmeditor (wieder gilt, daß die Dateien nur aus ASCII-Werten bestehen dürfen) und geben Sie das Programm mit der Prüfsumme ein. Die eckigen Klammern lassen Sie weg und auch die Zeilennummer. Listing 6 zeigt, wie die Datei mit dem Programm aus Listing 2 aussehen muß. Listing 7 demonstriert das Ganze für das Basic-Programm aus Listing 4.

SCheck starten Sie jetzt mit

```
scheck <Name.Ext> /a
```

Den Schrägstrich »/« können Sie auch durch » – « ersetzen. Desgleichen darf die Option auch als Großbuchstabe angegeben werden. Der Checksummer verwandelt das Programm nun in eine Quelldatei für den entsprechenden Compiler. Dazu werden die ersten vier Zeichen abgeschnitten, die Prüfsumme berechnet und bei einem Fehler die entsprechende Zeile ausgegeben. Der Editor steht dann wieder zur Verfügung. An der Art, wie die Datei verändert wird, sehen Sie, daß hinter der Prüfsumme (ohne Klammern eingegeben) immer ein Leerzeichen stehen muß. Danach folgt der Programmtext (nur in Basic mit Zeilennummer).

Nach dem Durchlauf von SCheck steht wieder das fehlerfreie Programm ohne Prüfsumme auf der Diskette. Und wieder gilt, daß Sie nicht mit dem Original Ihrer stundenlangen Eintipparbeit arbeiten sollten, sondern nur mit einer Kopie derselben.

SCheck verfügt aber noch über eine weitere Option. Die Hexdump-Listings können Sie jetzt nämlich auch ohne den Debugger eingeben. Sie tippen ein Hexdum-Listing wieder mit dem Editor ein und starten dann SCheck ohne Option. Nach

```
scheck <Name.Ext>
```

wird der Text wie im ersten Fall zeilenweise mit Prüfsumme ausgegeben. Der Editor steht wiederum zur Korrektur zur Verfügung.

Befindet sich das Programm korrekt auf der Diskette, so wird SCheck mit

```
scheck <Name.Ext> /b
```

erneut gestartet. Für die Option gilt das schon oben Gesagte. Auf der Diskette wird nun eine Binärdatei mit der Erweiterung ».bin« erzeugt. Für direkt ausführbare Com-Dateien müssen Sie das Programm nur noch mit

```
ren <Name>.bin <Name>.com
```

umbenennen.

Doch auch bei dieser Eingabe werden Sie durch Prüfsummen direkt unterstützt. Geben Sie den Programmtext mit Prüfsumme ein und starten Sie den Checksummer mit

```
scheck <Name.Ext> /a
```

so werden die eingegebenen Daten zeilenweise überprüft. Die so erzeugte Datei entspricht dem Hexdump ohne Prüfsumme.

```
scheck <Name.Ext> /b
```

erzeugt die entsprechende Binärdatei.

Mit SCheck wird das Eingeben der Programme aus der DOS International bedeutend komfortabler und weniger fehlerträchtig. Wer allerdings kein Turbo Pascal besitzt, um das Programm zu starten, der findet auf der Databox zu der DOS Extra 2 und auf den Databoxen der DOS International ab der Ausgabe 3'88 das Programm als lauffähige Com-Datei. Ihren Listingeingaben steht nichts mehr im Wege.

(ev/hg)

```
program check;

{ Ultra-Check-Utility von DOS International }

{ Volker Everts, 10.1.88    }
{ Version 1.31              }
{ Letzte Änderung: 20.1.88 }


const
  ZEICHEN : string[32] = '0123456789ABCDEFGHKLMNPRSTUVWXYZ';
  SPACE = #32;
  BS    = #8;
  TAB   = #9;
  BEL   = #7;
  CR    = #13;
  EMPTY = '';
  ESC   = #27;

type
  ZeilenTyp = string[255];

var
  Zeile: ZeilenTyp;
  ps, check: string[5];
  Name, TempName, Option: string[40];
  Ext: string[4];
  t, temp: text;
  f: file of byte;
  Nummer, b, code: integer;
  by: byte;
  ch: char;
  weiter, Auto, bin: boolean;


procedure init;

var
  i: integer;

begin
  Auto := false;
  bin := false;
  window(1,1,80,25);
  ClrScr; writeln;
  writeln('╔══════════════════════════════════════════════════════════════════════╗');
  writeln('║                                   ***    SCheck 1.31    ***                    ║');
  writeln('║                                                                                  ║');
  writeln('║                                                                                  ║');
  writeln('╠══════════════════════════════════════════════════════════════════════╗');
  writeln('║                                                                                  ║');
  writeln('║                                                                                  ║');
  writeln('║                                                                                  ║');
  writeln('║                                                                                  ║');
  writeln('║                                                                                  ║');
  writeln('╠══════════════════════════════════════════════════════════════════════╗');
  writeln('║ CHECK:                    ZEILE:                                                 ║');
  writeln('╚══════════════════════════════════════════════════════════════════════╝');
  GotoXY(1,20);
  writeln('┌──────────────────────────────────────────────────────────────────────┐');
  writeln('│                                                                                  │');
  writeln('└──────────────────────────────────────────────────────────────────────┘');
  if ParamCount <> 0 then
    for i := 1 to ParamCount do
      begin
        Option := ParamStr(i);
        if (Option = '-a')
        or (Option = '-A')
        or (Option = '/a')
        or (Option = '/A') then Auto := true
        else
        if (Option = '-b')
        or (Option = '-B')
        or (Option = '/b')
        or (Option = '/B') then bin := true
        else Name := Option;
      end
    else Name := EMPTY;
  repeat
    if Name = EMPTY then
      begin
        GotoXY(3,21);
        write('Geben Sie einen Dateinamen ein');
        GotoXY(54,12);
        BufLen := 24;
        read(Name);
        GotoXY(3,21);
        write('                                                        ');
      end;
    assign(t, Name);
    {$I-} reset(t); {$I+}
    if IOresult > 0 then
      begin
        for i := 1 to 3 do
          begin
            GotoXY(10,21);
            write('+++++  D A T E I F E H L E R +++++', BEL);
            delay(200);
            GotoXY(10,21);
            write('                                                        ');
            delay(200);
          end;
        Name := EMPTY;
        GotoXY(54,12);
        write('                                                        ');
      end;
  until Name <> EMPTY;
  for i := 1 to length(Name) do Name[i] := UpCase(Name[i]);
  i := pos('.',Name);
  if i = 0 then
    begin
      Name := Name + '.';
      i := length(Name);
    end;
  Ext := copy(Name, i, 4);
  Name := copy(Name, 1, i-1);
  GotoXY(54,12);
  write(Name, Ext);
  TempName := Name + '.$$$';
  assign(temp, TempName);
  rewrite(temp);
  if bin then
    begin
      assign(f, Name + '.BIN');
      rewrite(f);
      auto := false;
    end;
  Nummer := 0;
end;
```

```
129:
130:
131: procedure Pruefe(var z: ZeilenTyp);
132:
133: { Prüfsumme berechnen. Führende Leerzeichen und T
    abulatoren }
134: { werden nicht berücksichtigt. Bei Steuerzeichen
    im Text    }
135: { erfolgt eine Warnung
           }
136:
137: var a, p, i, j: integer;
138:
139: begin
140:   p := 0;
141:   a := 1;
142:   while (z[a] = SPACE) or (z[a] = TAB) do a := a
    + 1;
143:   for i := a to length(z) do
144:     if z[i] < #32 then
145:       for j := 1 to 3 do
146:         begin
147:           GotoXY(3,21);
148:           write(' +++++ WARNUNG: Steuerzeichen in
     Zeile !              ', BEL);
149:           delay(200);
150:           GotoXY(3,21);
151:           write('
           ');
152:           delay(200);
153:         end
154:     else p := ((i-a+1) and 7) * ord(z[i]) + p;
155:   ps := '    ';
156:   ps[1] := '[';
157:   for i := 2 to 4 do
158:     begin
159:       ps[i] := ZEICHEN[(p and 31) + 1 ];
160:       p := p shr 5;
161:     end;
162:   ps[5] := ']';
163: end;
164:
165:
166: procedure edit(var z: ZeilenTyp);
167:
168: const s = #32;
169:
170: var
171:   x: integer;
172:   ch: char;
173:
174: procedure schreibe(c: char);
175: begin
176:   if length(z) = 255 then write(BEL)
177:   else
178:     begin
179:       insert(c, z, x);
180:       x := x + 1;
181:       GotoXY(1,1);
182:       write(z);
183:     end;
184: end;
185:
186: procedure rechts;
187: begin
188:   if x > length(z) then write(BEL)
189:   else x := x + 1;
190: end;
191:
192: procedure links;
193: begin
194:   if x = 1 then write(BEL)
195:   else x := x - 1;
196: end;
197:
198: procedure hoch;
199: begin
200:   if x < 78 then write(BEL)
201:   else x := x - 77;
202: end;
203:
204: procedure runter;
205: begin
206:   if x + 77 > length(z) then write(BEL)
207:   else x := x + 77;
208: end;
209:
210: procedure entf;
211: begin
212:   if x <= length(z) then
213:     begin
214:       delete(z, x, 1);
215:       GotoXY(1,1);
216:       write(z,s);
217:     end
218:   else write(BEL);
219: end;
```

```
220:
221: procedure del;
222: begin
223:   if x = 1 then write(BEL)
224:   else
225:     begin
226:       delete(z, x-1, 1);
227:       x := x - 1;
228:       GotoXY(1,1);
229:       write(z,s);
230:     end;
231: end;
232:
233: procedure pos1;
234: begin
235:   x := 1;
236: end;
237:
238: procedure ende;
239: begin
240:   x := length(z) + 1;
241: end;
242:
243: begin
244:   GotoXY(3,21);
245:   write(' Editieren Sie die Programmzeile
                          [RETURN] Ende');
246:   window(2,6,78,10);
247:   ClrScr;
248:   write(z);
249:   GotoXY(1,1);
250:   x := 1;
251:   repeat
252:     read(kbd, ch);
253:     case ch of
254:       ESC:
255:         begin
256:           read(kbd, ch);
257:           case ch of
258:             'M': rechts;
259:             'K': links;
260:             'H': hoch;
261:             'P': runter;
262:             'G': pos1;
263:             'O': ende;
264:             'S': entf;
265:           end;
266:         end;
267:       BS: del;
268:       CR: ;
269:       else schreibe(ch);
270:     end;
271:   GotoXY(x mod 77, x div 77 + 1);
272:   until ch = CR;
273: end;
274:
275:
276: procedure ZeigeZeile;
277: begin
278:   if Auto then
279:     begin
280:       check := '[' + copy(Zeile, 1, 3) + ']';
281:       Zeile := copy(Zeile, 4, 255);
282:     end;
283:   window(2,6,78,10);
284:   ClrScr;
285:   write(Zeile);
286:   window(1,1,80,25);
287:   GotoXY(29,12);
288:   write(Nummer);
289:   pruefe(Zeile);
290:   GotoXY(10,12);
291:   write(ps);
292:   if bin then exit;
293:   if Auto then
294:      if ps = check then
295:        begin
296:          GotoXY(3,21);
297:          write(' Prüfsumme OK für Zeile ', Nummer
    );
298:          weiter := true;
299:        end
300:      else
301:        begin
302:          GotoXY(3,21);
303:          write(BEL, ' Prüfsummenfehler !    [E] E
    ditieren   [RETURN] Weiter   [Ctrl-C] Abbrechen');
304:          GotoXY(15,12);
305:        end
306:   else
307:     begin
308:       GotoXY(3,21);
309:       write(' [E] Editieren      [RETURN] Weiter
       [Q] Quitt      [Ctrl-C] Abbrechen');
310:       GotoXY(15,12);
311:     end;
312: end;
```

```
313:
314:
315: procedure Beenden;
316: var fi: file;
317: begin
318:   while not eof(t) do
319:     begin
320:       readln(t, Zeile);
321:       writeln(temp, Zeile);
322:     end;
323:   close(t); close(temp);
324:   if bin then close(f);
325:   assign(fi, Name + '.BAK');
326:   {$I-} reset(fi); {$I+}
327:   if IOresult = 0 then erase(fi);
328:   rename(t, Name + '.BAK');
329:   rename(temp, Name + Ext);
330:   GotoXY(3,21);
331:   if bin then Ext := '.BIN';
332:   write(' Die Datei ', Name, Ext, ' wurde gespeic
    hert ');
333:   while WhereX < 78 do write(' ');
334:   GotoXY(1,25);
335:   halt;
336: end;
337:
338:
339: procedure Quitt;
340: begin
341:   GotoXY(3,21);
342:   write(' Drücken Sie [J], um die Arbeit zu beend
    en :                                    ');
343:   GotoXY(48,21);
344:   read(kbd,ch);
345:   write(ch);
346:   if (ch = 'j') or (ch = 'J') then
347:     begin
348:       writeln(temp, Zeile);
349:       Beenden;
350:     end;
351: end;
352:
353:
354: begin
355:   init;
356:   if bin then
357:     begin
358:       GotoXY(3,21);
359:       write(' >>> Binärdatei wird erzeugt <<<
    ');
360:     end;
361:   while not eof(t) do
362:     begin
363:       readln(t, Zeile);
364:       repeat
365:         weiter := false;
366:         ZeigeZeile;
367:         if bin then weiter := true;
368:         if not weiter then
369:           begin
370:             read(kbd, ch);
371:             case ch of
372:               CR : weiter := true;
373:               'e','E': Edit(Zeile);
374:               'q','Q': if not Auto then Quitt;
375:             end;
376:           end;
377:       until weiter;
378:       writeln(temp, Zeile);
379:       if bin then
380:         begin
381:           while copy(Zeile, 1, 1) = SPACE do dele
    te(Zeile, 1, 1);
382:           repeat
383:             val('$' + copy(Zeile, 1, 2), b, code)
    ;
384:             if (code > 0) or (b < 0) or (b > 255)
     then
385:               begin
386:                 GotoXY(3,21);
387:                 write(' +++++ ABBRUCH WEGEN DATEN
    FEHLER +++++                                    ');
388:                 GotoXY(1,25);
389:                 halt;
390:               end;
391:             by := b;
392:             write(f, by);
393:             Zeile := copy(Zeile, 3, 255);
394:             while copy(Zeile, 1, 1) = SPACE do de
    lete(Zeile, 1, 1);
395:           until Zeile = EMPTY;
396:         end;
397:       Nummer := Nummer + 1;
398:     end;
399:   Beenden;
400: end.
```

Listing 1. SCheck prüft jede Zeile auf Fehler

```
[V16]    0:   VERSION = 'Version 2.3';
[H42]    1:   SPACE = #32;
[2G2]    2:   SPACE5 = '     ';
[Y3E]    3:   ZEICHEN : string[32] = '0123456789ABCDEFG
               HKLMNPRSTUVWXYZ';
[EMO]    4: var
[F1N]    5:   Nummern, Drucker, Elite, Schmal, Fett, Tx
               t, Pruef, HexDump: boolean;
[FEL]    6:   Summe, Breite, Hoehe, ZeilenNummer, Druck
               Zeile, Zaehler: integer;
[RG2]    7:   p, t: text;
[MN2]    8:   f: BinFile;
[FB2]    9:   LZ: char;
[PM2]   10:   b: byte;
[YB5]   11: procedure info;
[BH1]   12: begin
[9Z2]   13:   writeln;
[EVD]   14:   writeln(BEL, '+++++ Fehler bei Parametera
               ngabe !');
```

Listing 2. So werden Pascal-Programme ab sofort gedruckt

```
  VERSION = 'Version 2.3';
  SPACE = #32;
  SPACE5 = '     ';
  ZEICHEN : string[32] = '0123456789ABCDEFGHKLMNPRSTUVWXYZ';
var
  Nummern, Drucker, Elite, Schmal, Fett, Txt, Pruef, HexDump: boolean;
  Summe, Breite, Hoehe, ZeilenNummer, DruckZeile, Zaehler: integer;
  p, t: text;
  f: BinFile;
  LZ: char;
  b: byte;
procedure info;
begin
  writeln;
  writeln(BEL, '+++++ Fehler bei Parameterangabe !');
```

Listing 3. So müssen Sie das Pascal-Listing eingeben

```
V16   VERSION = 'Version 2.3';
H42   SPACE = #32;
2G2   SPACE5 = '     ';
Y3E   ZEICHEN : string[32] = '0123456789ABCDEFGHKLMNPRSTUVWXYZ';
EMO var
F1N   Nummern, Drucker, Elite, Schmal, Fett, Txt, Pruef, HexDump: bool
FEL   Summe, Breite, Hoehe, ZeilenNummer, DruckZeile, Zaehler: integer
RG2   p, t: text;
MN2   f: BinFile;
FB2   LZ: char;
PM2   b: byte;
YB5 procedure info;
BH1 begin
9Z2   writeln;
EVD   writeln(BEL, '+++++ Fehler bei Parameterangabe !');
```

Listing 4. Pascal-Programme mit Prüfsumme eingegeben

```
[PRC] 1010 SCREEN 9:KEY OFF:CLS:COLOR 1,0:PALETTE 1,7:P
      ALETTE 7,1
[6U4] 1020 COLORSW=1:GOTO 1100
[213] 1025 RESUME 1030
[H3B] 1030 ON ERROR GOTO 0:SCREEN 2:KEY OFF:CLS:COLORSW
      =0
[164] 1040 VY1=199:VY2=150
[CDK] 1100 LOCATE 10,1:PRINT "Please type the longitude
       of your country"
```

Listing 5. Bei der Eingabe müssen Sie auf die Zeilennummer in eckigen Klammern verzichten

```
1010 SCREEN 9:KEY OFF:CLS:COLOR 1,0:PALETTE 1,7:PALETTE 7,1
1020 COLORSW=1:GOTO 1100
1025 RESUME 1030
1030 ON ERROR GOTO 0:SCREEN 2:KEY OFF:CLS:COLORSW=0
1040 VY1=199:VY2=150
1100 LOCATE 10,1:PRINT "Please type the longitude of your country"
```

Listing 6. Basic-Programme sehen ab sofort in der DOS so aus

```
PRC 1010 SCREEN 9:KEY OFF:CLS:COLOR 1,0:PALETTE 1,7:PALETTE 7,1
6U4 1020 COLORSW=1:GOTO 1100
213 1025 RESUME 1030
H3B 1030 ON ERROR GOTO 0:SCREEN 2:KEY OFF:CLS:COLORSW=0
164 1040 VY1=199:VY2=150
CDK 1100 LOCATE 10,1:PRINT "Please type the longitude of your country"
```

Listing 7. Auch Basic-Programme können mit Prüfsumme eingegeben werden

CAD-Programm selbst programmiert

Einen Höhepunkt für Grafiken mit einem Personalcomputer stellen die verschiedenen CAD-Programme dar. Normalerweise ist diese Software teuer – doch nicht SCad, das Sie hier finden. Das Programm ist völlig in Turbo Pascal geschrieben, so daß Sie es für eigene Zwecke sehr einfach erweitern können.

Dreidimensionale Grafiken zu programmieren, ist eine Wissenschaft für sich. Es geht darum, einem Computer etwas »begreiflich« zu machen, was sich aus dem menschlichen Sehen ableitet. Mit Hilfe von Geometrie und physikalischen Gesetzen (Optik) müssen die Datenstrukturen räumlicher Objekte gespeichert und diese Körper dann auf eine Ebene (in den zweidimensionalen Raum) projiziert werden. Wenn das gelungen ist, ist es nicht mehr schwer, auch Objekte aus dem vierdimensionalen Raum darzustellen. Allerdings ist für deren Wiedergabe ein dreidimensionaler Monitor – oder das Bild eines Hologramms – sinnvoll. Für das Vorstellungsvermögen eines Menschen bleibt diese Dimension abstrakt.
Das Programm 3D-CAD macht es möglich, beliebige dreidimensionale Objekte zu zeichnen. Diese können gespeichert, editiert und modifiziert werden. Sie lassen sich desweiteren beliebig drehen und mit verschiedenen virtuellen Objektiven betrachten. Aufgrund der Datenmenge überfordert die Behandlung verdeckter Linien Ihren PC. Sie fallen deshalb weg.
Als erstes definieren Sie einen Raum, am einfachsten über dem Nullpunkt. Von diesem Ursprung aus erstrecken sich die Achsen in sechs Richtungen. Von der Schule her kennen Sie zwar auch andere Modelle, aber die Berechnung um einen Punkt herum eignet sich für die Arbeit mit einem Computer sehr gut.
Als nächstes folgt eine Punktliste, wobei ein Punkt mit Hilfe von drei Koordinaten im Raum (x,y und z) definiert wird. Jedem Punkt entspricht damit ein Feld von drei Integerzahlen. Das funktioniert natürlich nur, solange der Raum innerhalb der Grenzen für Integerzahlen (zwischen -65536 und 65535) liegt. In allen anderen Fällen müssen Sie sich mit Realzahlen behelfen, wovon allerdings aus Speicherplatzgründen abgeraten werden muß.
Als nächstes benötigen Sie eine Liste für die Kanten des abzubildenen Körpers. Das ist leicht zu realisieren, denn dazu sind nur zwei Informationen erforderlich: die Position der beiden zu verbindenden Punkte und die Option, ob die Verbindungslinie der Punkte auf dem Bildschirm gezeichnet werden soll. Da Kanten in fortlaufender Folge ansteigen, genügt für die Definition, ob sie gezeichnet werden sollen oder nicht, allein das Vorzeichen.
Schwieriger wird es mit den verdeckten Linien. Ein Computer kann nämlich beim besten Willen nicht sehen. Es wären weitere Informationen notwendig. Man bräuchte eine Flächenliste, die jeden Punkt, der zu einer Fläche gehört, aufnimmt. Und das ist noch nicht alles: Ein Würfel, der vollkommen geschlossen ist, hat dieselbe Punkt- und Kantenstruktur wie ein gleich großer Kasten, der an einer Seite offen ist (dem also der »Deckel« fehlt). Sie brauchen also noch eine Variable, die die Auskunft gibt, ob eine Fläche zu sehen ist.
Prinzipiell stehen Sie hier vor dem gleichen Problem wie bei den Kanten, nur benötigt jetzt der ganze Vorgang bedeutend mehr Speicherplatz. Die größte Schwierigkeit liegt bei den nur teilweise sichtbaren Kanten. Hier können Sie sich nicht mehr mit Integerwerten behelfen, sondern müssen auf die Vektorrechnung zurückgreifen. In dem hier vorgestellten Programm ist diese aber nicht integriert, so daß solche Feinheiten nicht berücksichtigt werden.
Das CAD-Programm wird mit Hilfe der Stapeldatei aus Listing 1 gestartet. Der Grund für diesen etwas umständlichen Start liegt in der Unfähigkeit von Turbo Pascal, das Directory formatiert auf den Bildschirm zu bringen. Somit legen wir als erstes die Informationen des Directories in der Datei »3ddat.dir« ab. Geben Sie mit der Batch-Datei gleich den Namen eines Objekts an (allerdings ohne die Namenserweiterung ».3d«), so wird das Bild automatisch geladen und steht zur weiteren Bearbeitung direkt zur Verfügung. Die Syntax lautet dann

```
scad062 <Objektname>
```

also beispielsweise

```
scad062 kugel
```

Nach dem Start meldet sich auf Tastendruck das Hauptmenü (Bild 1). Sie erkennen fünf verschiedene Fenster. Oben links ist die Versionsnummer des Programms sowie der Name der im Moment bearbeiteten Objekte eingeblendet. Rechts daneben finden Sie das Optionen-Fenster. Die großen, hell dargestellten Buchstaben machen die Buchstabentaste kenntlich, die jeweils die entsprechende Funktion aufruft. Welche Funktionen zur Verfügung stehen, finden Sie in Bild 2. Direkt darunter wird die Punktzahl, Eckenzahl sowie die Zahl der Einzelobjekte angezeigt. Das große Fenster dient dem Dialog, im unteren Fenster finden Sie die Tasten dargestellt, mit denen die Perspektive verändert wird. Auch aktuelle Parameter, wie beispielsweise Rotationswinkel und so weiter, werden hier angezeigt.

Es gibt kaum eine Grafik, die Sie mit diesem SCad nicht auf Ihren Bildschirm zeichnen können

SCad besitzt folgende Funktionen:
? – wechselt das Directory.
/ – rechnet (nicht zu große) Objekte in eine fortlaufende Bildfolge um. Der entstehende Film wird nach der Rechenphase auf dem Bildschirm gezeigt. Durch einen Druck auf die Esc-Taste gelangt man zurück ins Hauptmenü.
B – zeigt das Objekt aus einer zufälligen Perspektive.
Während der Bearbeitung des Unterprogramms stehen folgende Optionen zur Verfügung:
[Esc] – führt zurück zum Hauptmenü.
[Enter] – führt ebenfalls zurück zum Hauptmenü, allerdings unter Beibehaltung der neuen Ansicht.
Mit den Pfeiltasten läßt sich das Objekt auf dem Bildschirm verschieben. Jede andere Taste wählt eine neue zufällige Projektion.
V – funktioniert nur bei einem bereits existierenden Objekt (»shapes« muß größer als Null sein). Die aktuelle Ansicht wird gezeigt. Auf Tastendruck gelangt man zurück ins Hauptmenü.
W – hat dieselbe Wirkung wie V. Hier werden allerdings die Punkte fortlaufend durchnumeriert. Diese Option ist sehr wichtig, um beim Aufbau des Bildes die Kontrolle

über die Daten zu behalten. Die Objekte lassen sich von allen Seiten betrachten – nicht nur unter den üblichen Winkeln von 90 und 180 Grad. Die Darstellung kann in allen drei Raumrichtungen um bis zu 360 Grad gedreht werden.

Y – ist ein Umschalter. Mit » – « wird das Objekt nach [V] oder [W] komplett aufgebaut. Bei » + « wird nach jeder Linie ein Tastendruck abgewartet, bevor das Bild weiter gezeichnet wird. [Esc] bricht die Routine ab. Mit diesem Unterprogramm kann ein Bild sehr langsam aufgebaut werden. Am oberen Rand des Bildes sind die zu jeder Kante gehörenden Daten angegeben. Somit können Sie die Werte während des Aufbaus kontrollieren.

S – dient zum Speichern von Objekten. Der Inhalt der Datei »3ddat.dir« wird in der Mitte des Bildschirmes ausgegeben. Die Dateinamen müssen Sie ohne Erweiterung angeben. Mit dieser Einschränkung läßt sich das interne Inhaltsverzeichnis leichter verwalten. Auch für diese Anweisung muß »shapes« größer als Null sein.

L – Objekte laden. Das Inhaltsverzeichnis wird in der Mitte des Bildschirms ausgegeben. Sie müssen den Namen des gewünschten Objekts eintippen. I/O-Fehler beim Zugriff auf den Datenträger werden zwar abgefangen, jedoch nur zum Teil dem Benutzer gemeldet. Hier bieten sich noch Verbesserungen an.

M – Objekte miteinander verknüpfen (dazuladen). Dieses Unterprogramm gibt Ihnen die Möglichkeit, Bilder in Einzelteilen zu erzeugen und erst zum Schluß zusammenzubinden. »shapes« muß wiederum größer als Null sein.

Mit dem Editor können Sie Grafiken zeichnen, ohne sich um die Punktwerte zu kümmern

E – ruft den Editor auf.

Punkte sind bei SCad durch die drei Integerwerte x,y und z vorgegeben. Eine Kante ist immer durch den Wert eines bestimmten Punkts festgelegt; hat die erste Kante den Wert 62 (Punkt 62) und die zweite Kante den Wert -102 (Punkt 102), so wird eine Linie vom Punkt 62 zum Punkt 102 gezeichnet. Hat die zweite Kante den Wert 102, so wird die Linie nicht gezeichnet. Damit eine Linie tatsächlich dargestellt wird, muß eine Kante einen positiven Wert haben. Negative Werte deuten auf Linien, die nur projeziert, aber nicht gezeichnet werden sollen.

Als Beispiel finden Sie hier die Datenstruktur eines Würfels mit 200 Einheiten Kantenlänge rund um den Ursprung (x = y = z = 0). Die Werte für die Punkte lauten:

Punkt 1 x = -100 y = -100 z = -100
Punkt 2 x = 100 y = -100 z = -100
Punkt 3 x = 100 y = 100 z = -100
Punkt 4 x = -100 y = 100 z = -100
Punkt 5 x = -100 y = -100 z = 100
Punkt 6 x = 100 y = -100 z = 100
Punkt 7 x = 100 y = 100 z = 100
Punkt 8 x = -100 y = 100 z = 100

Die Daten für die Kanten lauten:

Kante 1 = -1 (zeichne nicht zu Punkt 1)
Kante 2 = 2 (zeichne zu Punkt 2)
Kante 3 = 3 (zeichne zu Punkt 3)
Kante 4 = 4 (zeichne zu Punkt 4)
Kante 5 = 1 (zeichne zu Punkt 1)
Kante 6 = -5 (zeichne nicht zu Punkt 5)
Kante 7 = 6 (zeichne zu Punkt 6)
Kante 8 = 7 (zeichne zu Punkt 7)
Kante 9 = 8 (zeichne zu Punkt 8)
Kante 10 = 5 (zeichne zu Punkt 5)
Kante 11 = -1 (zeichne nicht zu Punkt 1)
Kante 12 = 5 (zeichne zu Punkt 5)
Kante 13 = -2 (zeichne nicht zu Punkt 2)
Kante 14 = 6 (zeichne zu Punkt 6)
Kante 15 = -3 (zeichne nicht zu Punkt 3)
Kante 16 = 7 (zeichne zu Punkt 7)
Kante 17 = -4 (zeichne nicht zu Punkt 4)
Kante 18 = 8 (zeichne zu Punkt 8)

Durch die negativen Vorzeichen ist also sichergestellt, die richtige Projektion der Linien zu erhalten und Punkte selektiv setzen und verbinden zu können.

Zurück zum Editor: Auf Druck der E-Taste wird im Fenster die Frage gestellt, wo die Eingabe der Punkte, die zu ändern oder neu einzugeben sind, beginnen soll. Es wird der momentane Wert des entsprechenden Punktes angezeigt. Nun kann der Anwender die Daten verändern. Das geht so lange Punkt für Punkt weiter, bis Sie die abschließende Bestätigung nicht mit [Enter] quittieren. Die folgende Frage nach der neuen Punktezahl bestätigen Sie nur mit [Enter], wenn die alten Punkte nicht erhalten bleiben sollen. Analog zu den Punkten werden danach die Kanten abgefragt.

! – funktioniert nur, wenn »shapes« größer als Null ist. Das Unterprogramm optimiert die Daten. Dazu können einzelne Punkte oder Kanten gelöscht werden. Beantworten Sie die Frage nach dem Optimieren mit [Enter], so geschieht folgendes:

Eine Routine sucht nach doppelten und verdeckten Punkten

Die Routine sucht den gesamten Datensatz nach doppelten oder unnötigen negativen Kanten ab. Doppelte oder unsichtbare Punkte werden entfernt. (Sie entstehen leicht beim Erzeugen von Rotationskörpern.)

P – verschiebt Objekte entlang der x-, y- und z-Achsen (also im Raum). Mit dem Add-Menü werden die Objekte miteinander verbunden (perspektivische Schriften und so weiter). Für diese Routine muß »shapes« größer als Null sein.

F – verformt ein Objekt in x-, y- und z-Richtung. Mit dem Add-Menü lassen sich die Objekte wiederum verbinden (beispielsweise Pyramiden und Säulen). »shapes« muß größer als Null sein. Auch die Eingabe von negativen Werten zwecks Spiegelung des Objekts an der entsprechenden Achse ist erlaubt. Verkleinerungen fordert SCad als Realwert an (zum Beispiel: 1/2 = 0.5).

T – dreht das Objekt. Der Drehwinkel ist als Integerwert in Grad anzugeben, als Drehachse fungiert die y-Achse. Wollen Sie das Objekt anders drehen, so müssen Sie zuvor mit [C] die Achsen entsprechend vertauschen. Auch für diese Routine muß »shapes« größer als Null sein.

A – erzeugt ein Objekt. Es wird ein zum aktuellen identi-

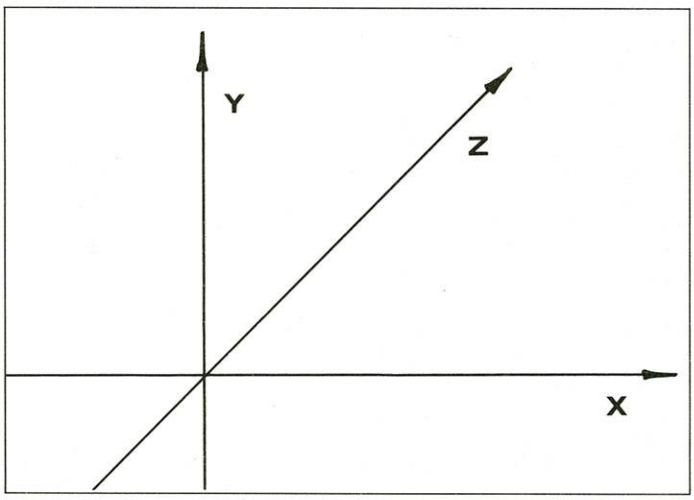


Bild 1. Drei Raumrichtungen gibt es

```
[BS4]     0: dir *.3d > 3ddat.dir
[CZO]     1: scad
```

Listing 1. Mit dieser Batch-Datei starten Sie das CAD-Prgramm

sches Objekt erzeugt. Dieses kann allerdings gleichzeitig verschoben, gedreht und deformiert werden, so daß das neue Objekt mit dem ursprünglichen nur noch die Zahl der Punkte und Kanten gemeinsam hat. Die Fragen für die Veränderungen (P,F,T,D und C auch in Kombination) stellt das Programm im Dialogfenster. Für »shapes« gilt wieder die Bedingung, größer als Null zu sein.

K – löscht ein, mehrere oder alle Objekte, sofern »shapes« größer als Null ist.

R – erzeugt einen Rotationskörper. Rotiert wird um die y-Achse. »Zahl der Linien« fragt die Kanten ab (3 bis 18). Bei 18 Kanten wird alle 20 Grad eine Linie gezeichnet. Teilrotationen werden als Realwerte angegeben (beispielsweise für einen halben Kreis »0.5«). »Linien verbinden« erzeugt einen massiven Rotationskörper.

Für einen Rotationskörper geben Sie eine gekrümmte Linie aus verschiedenen x- und y-Werten ein. z kann Null sein, muß es aber nicht. An Stelle einer Linie lassen sich aber auch Körper rotieren. Drücken Sie bei »Linien verbinden« nicht [Enter], sondern eine andere Taste, so verteilen Sie identische Objekte gleichmäßig um die y-Achse herum. Dabei werden so viele Körper gezeichnet, wie Sie bei Zahl der Linien angegeben haben. Auch diese Routine erfordert »shapes« größer als Null.

D – deformiert das Objekt in Abhängigkeit von einer Zufallszahl. Das Gebirge aus Bild 3 entstand mit dieser Funktion. »shapes« muß wieder größer als Null sein.

U – »schleust« die Objekte durch eine Auswahl von trigonometrischen Funktionen. Vor dem Ausprobieren sollten Sie Ihre Daten unbedingt speichern. »shapes« muß wieder größer als Null sein.

H – ist ein Schalter. Mit » + « werden zusätzliche Objekte vom Ursprungsobjekt farblich getrennt, mit » – « unterbleibt dies. Damit können Sie sich die Arbeit mit der K-Routine erleichtern.

C – tauscht die Raumachsen. Sind sie im Add-Menü über Kreuz gelegt, entstehen keine Spiegelungen. Das Objekt bleibt, wie es war. »shapes« muß größer als Null sein.

X – ist wiederum ein Schalter. Mit » + « werden die Raumachsen und der Ursprung gezeichnet, » – « unterbindet dies. Ohne diesen Schalter auf » + « zu setzen, lassen sich keine Objekte erzeugen. Die Ausdehnung des Raumkreuzes erstreckt sich über 1600 Einheiten (jeweils von -800 bis + 800). Es können jedoch auch Objekte mit größerer Ausdehnung eingegeben werden.

Q – beendet das Programm.

Einfache Körper werden automatisch erzeugt – Sie brauchen also gar nichts dazu zu tun

J – Das Programm kann einige einfache Körper automatisch erzeugen (Würfel, Tetraeder und Oktaeder). [J] ruft die entsprechende Routine auf. Sie haben die Wahl, ob das Objekt vom Zufallsgenerator gezeichnet werden soll, oder ob Sie es genau definieren wollen. Bei der Definition wird die Größe und die Lage des Schwerpunkts (bei diesen Körpern immer in der Mitte des Körpers) im Raum abgefragt.

N – verbindet alle Punkte in fortlaufender Folge. Die Routine hilft beim Aufspüren unsichtbarer Punkte und dem Erkennen der Datenstruktur. »shapes« muß wieder größer als Null sein. Vor dem Aufruf sollten Sie unbedingt Ihre Daten sicherheitshalber speichern.

O – ändert auf Knopfdruck die Brennweite des virtuellen Objektivs. Je größer die Zahl ist, desto kleiner ist die Brennweite. Nach dem größten Wert schaltet der Computer automatisch auf 1 zurück. Bei extremen Perspektiven kommt es zu Divisionen annähernd Null. Die Folgen sind nicht immer vorhersehbar. Aus diesem Grund sollten Sie vor dem Aufruf dieser Routine auch hier Ihre Daten speichern.

G – erlaubt die Eingabe einer bestimmten Projektion. Blickwinkel, x, y und z müssen als Integerzahlen (Grad) eingegeben werden.

\+ – vergrößert die Abbildung.

\- – verkleinert die Abbildung.

Die Projektion wird bei den beiden letzten Routinen beibehalten. Mit den Unterprogrammen lassen sich beim Editieren Teile vergrößern, ohne daß das Objekt mit [F] verändert wird.

Z – setzt die Abbildungsgröße zurück auf Null.

I – setzt den Ursprung (x = y = z = 0) wieder in die Bildmitte, falls er beim Wechsel der Ansicht ([B] und dann Pfeiltasten) verschoben wurde.

* – druckt alle Punkt- und Kantendaten zur Kontrolle auf dem Drucker aus.

Das Bild selbst wird mit Hilfe der Taste [PrtSc] gedruckt. In der Startdatei (Listing 1) fügen Sie an den Anfang die Zeile

```
graphics/r
```

oder einen anderen Druckertreiber ein. Mit [V] wird das Bild aufgebaut. Schalten Sie mit [V] die Achsen unsicht-

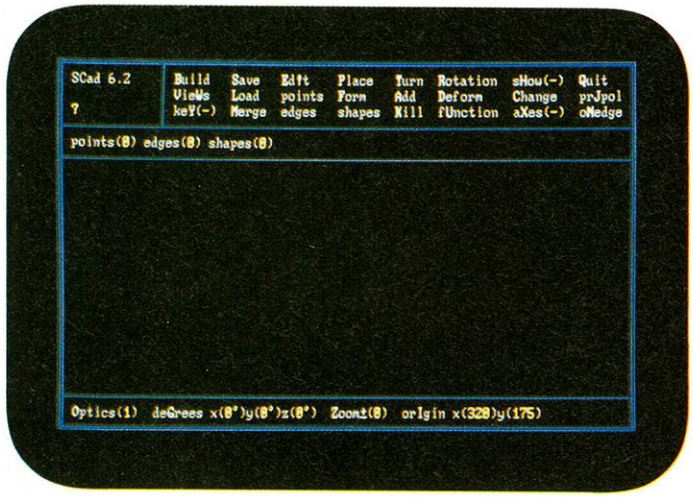


Bild 2. Das Hauptmenü von »SCad«

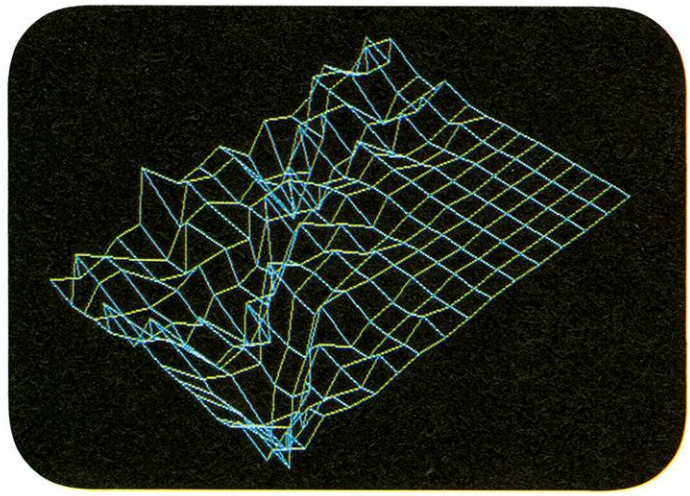


Bild 3. Das Gebirge wurde mit der D-Option erzeugt

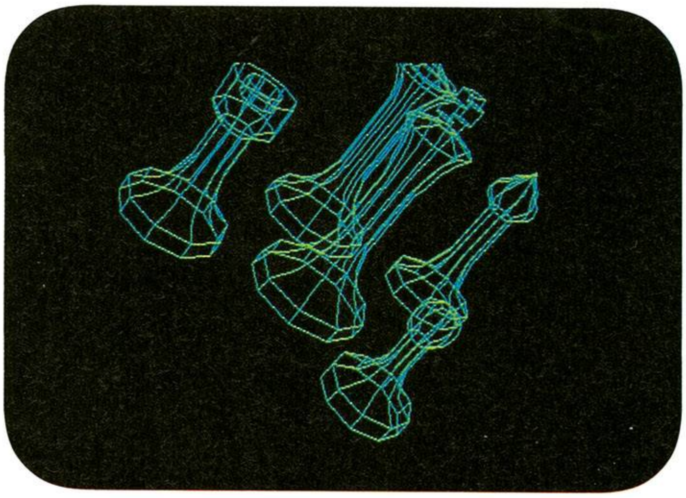


Bild 4. Auch die Schachfiguren entstehen aus einfachen Grundstrukturen

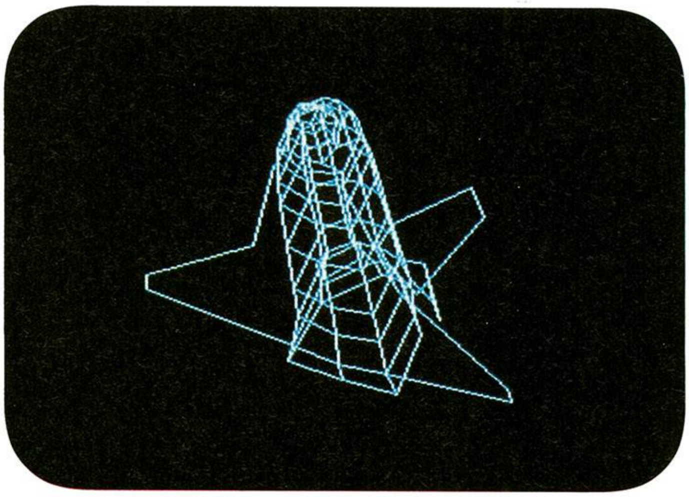


Bild 5. Auch ein Space Shuttle läßt sich mit SCad zeichnen

bar und das Bild kann ausgegeben werden. Jetzt stehen Sie endlich vor der Frage, wie denn nun ein Objekt überhaupt eingegeben wird? Sie können zum einen das Objekt direkt mit dem Programm erzeugen. Falls Sie aber nicht ständig am Computer sitzen wollen, dann ist der einfachere Weg, das gewünschte Objekt erst mit dem Bleistift auf Millimeterpapier zu skizzieren und die Punkte – beziehungsweise ihre drei Werte dahinter – inklusive Verbindungskanten durch Ziffern einzugeben. Modifikationen und die Feinarbeit führen Sie mit den Routinen des Programms aus.

Natürlich gibt es auch andere Wege, um interessante Bilder zu erzeugen. Mit einem Netz in x- und z-Richtung (y = 0) lassen sich mit Hilfe der U-Option mehrere dreidimensionale Funktionen von allen Seiten betrachten. Mit einem grundlegenden räumlichen Vorstellungsvermögen werden die Objekte auf einfache Grundformen zurückgeführt. Auch komplizierteste Objekte entstehen so auf dem Bildschirm, wie zum Beispiel die Schachfiguren aus Bild 3.

SCad läuft mit einer EGA-Karte. Für andere Grafikkarten müssen Sie einige wenige Punkte entsprechend anpassen. Die Maximalwerte für die Ausdehnung des Projektionsbereichs (Bildschirmauflösung) in der Prozedur »pro2d« in den Zeilen, in denen u1, u2, v1 und v2 berechnet werden, müssen an die Werte der Grafikkarte angeglichen werden. In der Routine müssen Sie gegebenenfalls einen anderen Entzerrungsfaktor angeben. In den Variablen v1, v2, u1 und u2 stehen jeweils die umgerechneten Bildschirmkoordinaten.

Im Programm treten folgende Werte auf, die speziell anzupassen sind:

»639« ist die maximale Auflösung in horizontaler Richtung bei der EGA-Karte. Für CGA bleibt sie gleich, für Hercules muß sie vergrößert werden.

»349« ist die maximale vertikale Auflösung für die hier verwendete EGA-Karte. CGA arbeitet mit 199 Punkten, wärend eine Hercules-Karte den gleichen Wert verlangt. Entsprechend bezeichnet »320« die horizontale Bildmitte und »170« die vertikale (immer die Hälfte der oberen Werte).

»8« bezeichnet die Breite für ASCII-Zeichen bei einer EGA-Karte. CGA benutzt den gleichen Wert, während Hercules mit neun Punkten arbeitet.

»14« ist die Höhe eines ASCII-Zeichens EGA. Hercules-Karten haben den gleichen Wert, während eine CGA-Karte maximal acht Punkte in der Höhe für ein Zeichen zur Verfügung stellt.

Diese Werte müssen Sie ersetzen, um das Programm mit Ihrer Karte lauffähig zu machen. Desweiteren sind die Start- beziehungsweise Resetwerte für die Variablen »ox« und »oy« (Ursprungspunkte) im Hauptprogramm und in der Prozedur »getorigin« anzupassen. Die Grafikroutinen »egamode«, »clega«, »plot« und »line« müssen komplett durch die zu den entsprechenden Grafik-Adaptern passenden ersetzt werden.

Für die Hercules-Karte können Sie die Turbo-Graphix-Toolbox einsetzen. Besser ist es allerdings, sich selbst ein möglichst kurzes Programm für die Grafikroutinen zu schreiben. Die Turbo-Graphix-Toolbox ist nämlich sehr lang und umständlich programmiert. Mit der CGA-Karte bekommen Sie kaum Probleme (»Hires«, »ClearScreen«, »Pset« und »Draw«). »gxpos« und »gypos« berechnen die Mitte eines Textzeichen an einer Grafikposition. Um hier einzugreifen, müssen Sie die Höhe und die Breite eines Textzeichens in Punkten kennen.

Falls Ihr Computer weniger als 640 KByte Speicherplatz hat, dann sollten Sie die Maximalwerte für die Zahl der Punkte und Kanten sowie die erlaubten Werte in der Animationsprozedur verringern. Ansonsten fallen keine weiteren Punkte an, die Sie extra ummodeln müßten. Jedoch ist das Programm aus Listing 2 jederzeit ausbaufähig und Ihrem Programmiergeist steht hier nichts im Wege.

(Stephan Schulze/hg)

```
[VV3]   PROGRAM  CADEGA;
[000]
[OD4]   CONST g              =0.0001;
[M72]         d              =120;
[2S2]         max            =6000;
[PDD]         memo           ='Not enough memory press
        any key';
[COB]         akey           ='Press any key to contin
       ue';
[US9]         okey           =' or any key to continue
       ';
[EY7]         esc            ='Press ESC to return';
[BUU]         lin2o          ='╓══════════════╤════════
       ═════════════════════════════════════════
        ════════════╖';
[NCS]         lin1o          ='╟--------------┴--------
       -----------------------------------------
        ------------╢';
[2ES]         lin1u          ='╟-----------------------
       -----------------------------------------
        --┬---------╢';
[MUU]         lin2u          ='╙═══════════════════════
       ═════════════════════════════════════════
        ════════════╜';
[UVA]         spc            ='


                 ';
```

```
[LRA]   14:       vers          ='SCad 6.2    by Stefan S
          chultze';
[MV4]   15:       dtyp          ='3DDAT.DIR';
[286]   16:       col           :ARRAY[0..15] OF BYTE
[HN6]   17:                     =(0,1,2,3,4,5,6,7,8,9,10,
          11,12,13,14,15);
[2R8]   18: TYPE  punkt         =ARRAY[1..3] OF INTEGER;
[ED7]   19:       vektor        =ARRAY[1..3] OF REAL;
[ZZ6]   20:       textstring    =STRING[80];
[9EC]   21: VAR   p                                    :AR
          RAY[1..max] OF punkt;
[2DA]   22:       e                                    :AR
          RAY[1..max] OF INTEGER;
[G18]   23:       daf                                  :FI
          LE OF INTEGER;
[A96]   24:       w                                    :pu
          nkt;
[7N6]   25:       h,f                                  :ve
          ktor;
[7Z8]   26:       savemode,ox,oy,zo,b,u1,v1,u2,v2,
[YGA]   27:       pn,en,sn,pn1,en1,sn1,pk,ek           :IN
          TEGER;
[1U3]   28:       d1,d2,x1,y1,z1,
[CV9]   29:       am,bm,cm,dm,em,fm,gm,hm,im           :RE
          AL;
[UC5]   30:       t                                    :CH
          AR;
[R98]   31:       nam,nam1                             :ST
          RING[8];
[VN9]   32:       ende,IOer,key,axs,shw                :BO
          OLEAN;
[1W5]   33:       akt                                  :BY
          TE;
[000]   34:
[H98]   35: PROCEDURE egamode(bo :BOOLEAN);
[EM8]   36: TYPE  regpack =RECORD CASE INTEGER OF
[RGC]   37:                     1:(ax,bx,cx,dx,bp,si,di,ds
          ,es,flags :INTEGER);
[VU8]   38:                     2:(al,ah,bl,bh,cl,ch,dl,dh
          :BYTE);
[TMO]   39:                   END;
[YO5]   40: VAR   regs :regpack;
[B21]   41: BEGIN
[ZT3]   42:   IF bo THEN BEGIN
[SU7]   43:     regs.ax:=$0F00; INTR($10,regs);
[TG5]   44:     savemode:=regs.al;
[8B7]   45:     regs.ax:=$0010; INTR($10,regs)
[3B3]   46:   END ELSE BEGIN
[FW2]   47:     regs.ah:=0;
[KY9]   48:     regs.ax:=savemode; INTR($10,regs)
[TMO]   49:   END;
[TMO]   50: END;
[000]   51:
[TM4]   52: PROCEDURE clega;
[9U8]   53: BEGIN FILLCHAR(MEM[$A000:0000],$7000,0); EN
          D;
[000]   54:
[P2B]   55: PROCEDURE plot(xe,ye :INTEGER; farbe16 :BYT
          E);
[Z65]   56: VAR offset  :INTEGER;
[WC4]   57:     bitmask :BYTE;
[B21]   58: BEGIN
[GBC]   59:   IF (xe<0) OR (xe>639) OR (ye<0) OR (ye>34
          9) THEN EXIT;
[8Z6]   60:   bitmask:=$80 SHR (xe AND 7);
[PN6]   61:   offset :=(ye*80)+(xe SHR 3);
[B13]   62:   PORT[$3CE]:=0;
[HW4]   63:   PORT[$3CF]:=farbe16;
[G13]   64:   PORT[$3CE]:=1;
[SC3]   65:   PORT[$3CF]:=$0F;
[L23]   66:   PORT[$3CE]:=8;
[U65]   67:   PORT[$3CF]:=bitmask;
[VX9]   68:   MEM[$A000:offset]:=MEM[$A000:offset];
[WG3]   69:   PORT[$3CF]:=$FF;
[G13]   70:   PORT[$3CE]:=1;
[AP2]   71:   PORT[$3CF]:=0
[TMO]   72: END;
[000]   73:
[V5E]   74: PROCEDURE line(xe1,ye1,xe2,ye2 :INTEGER; fa
          rbe16 :BYTE);
[ZZB]   75: VAR   xe,ye,ze,dxe,dye,dze,num1,num2 :INTEG
          ER;
[B21]   76: BEGIN
[3L9]   77:   dxe:=ABS(xe1-xe2); dye:=ABS(ye1-ye2);
[4Y4]   78:   IF xe1<xe2 THEN BEGIN
[ZS4]   79:     xe:=xe1; ye:=ye1;
[PM7]   80:     IF ye1>ye2 THEN ze:=-1 ELSE ze:=1;
[3B3]   81:   END ELSE BEGIN
[6T4]   82:     xe:=xe2; ye:=ye2;
[UM7]   83:     IF ye2>ye1 THEN ze:=-1 ELSE ze:=1;
[TMO]   84:   END;
[WMA]   85:   IF dxe>dye THEN num2:=dxe ELSE num2:=dye;
[2V5]   86:   plot(xe,ye,farbe16);
[EDC]   87:   IF (dxe=0) OR (dye=0) THEN FOR num1:=1 TO
          num2 DO BEGIN
[9EB]   88:     IF dxe=0 THEN ye:=ye+ze ELSE xe:=SUCC(x
          e);
[2V5]   89:     plot(xe,ye,farbe16);
[3B3]   90:   END ELSE BEGIN
[N44]   91:     dze:=num2 DIV 2;
[RD6]   92:     FOR num1:=1 TO num2 DO BEGIN
[FYD]   93:       IF dze<dxe THEN BEGIN dze:=dze+dye; x
          e:=SUCC(xe) END;
[LLD]   94:       IF dze>=dxe THEN BEGIN dze:=dze-dxe;
          ye:=ye+ze END;
[2V5]   95:       plot(xe,ye,farbe16);
[TMO]   96:     END;
[TMO]   97:   END;
[TMO]   98: END;
[000]   99:
[S18]  100: PROCEDURE cursor(bo :BOOLEAN);
[F2K]  101: TYPE  regpack =RECORD ax,bx,cx,dx,bp,si,di,
          ds,es,flags :INTEGER END;
[YO5]  102: VAR   regs :regpack;
[B21]  103: BEGIN
[URF]  104:   WITH regs DO BEGIN ax:=$0100; IF bo THEN
          cx:=$0607 ELSE cx:=$2000 END;
[VW2]  105:   INTR(16,regs)
[TMO]  106: END;
[000]  107:
[6DM]  108: PROCEDURE text(farbtext :textstring; spalte
          ,zeile,farbe16 :BYTE);
[WKK]  109: BEGIN TEXTCOLOR(farbe16); GOTOXY(spalte,zei
          le); WRITE(farbtext) END;
[000]  110:
[ENE]  111: PROCEDURE txt(farbtext :textstring; farbe16
          :BYTE);
[NDC]  112: BEGIN TEXTCOLOR(farbe16); WRITE(farbtext) E
          ND;
[000]  113:
[ET7]  114: PROCEDURE wrt(iwert :INTEGER);
[SMA]  115: BEGIN TEXTCOLOR(col[14]); WRITE(iwert) END;
[000]  116:
[G5D]  117: PROCEDURE wert(iwert :INTEGER; spalte,zeile
          :BYTE);
[RBH]  118: BEGIN TEXTCOLOR(col[15]); GOTOXY(spalte,zei
          le); WRITE(iwert) END;
[000]  119:
[1W4]  120: PROCEDURE winon;
[3C5]  121: BEGIN WINDOW(3,8,78,21) END;
[000]  122:
[4W4]  123: PROCEDURE winoff;
[TB5]  124: BEGIN WINDOW(1,1,80,25) END;
[000]  125:
[OS7]  126: PROCEDURE pkt(x,y,z :INTEGER);
[B21]  127: BEGIN
[H6L]  128:   IF (pn<max) THEN BEGIN pn:=pn+1; p[pn][1]
          :=x; p[pn][2]:=y; p[pn][3]:=z END
[TMO]  129: END;
[000]  130:
[2F6]  131: PROCEDURE eck(i :INTEGER);
[PRD]  132: BEGIN IF (en<max) THEN BEGIN en:=en+1; e[en
          ]:=i END END;
[000]  133:
[3GA]  134: PROCEDURE loeschfeld; VAR i,j :INTEGER;
[B21]  135: BEGIN
[ALA]  136:   FOR i:=pn+1 TO max DO FOR j:=1 TO 3 DO p[
          i][j]:=0;
[9M6]  137:   FOR i:=en+1 TO max DO e[i]:=0;
[TMO]  138: END;
[000]  139:
[9VC]  140: FUNCTION gxpos(i :INTEGER) :INTEGER; VAR n
          :INTEGER;
[B21]  141: BEGIN
[2YE]  142:   n:=ROUND((i+4)/8); IF (n>80) THEN n:=80;
          IF (n<1) THEN n:=1; gxpos:=n
[TMO]  143: END;
[000]  144:
[CVC]  145: FUNCTION gypos(i :INTEGER) :INTEGER; VAR n
          :INTEGER;
[B21]  146: BEGIN
[XDF]  147:   n:=ROUND((i+7)/14); IF (n>25) THEN n:=25;
          IF (n<1) THEN n:=1; gypos:=n
[TMO]  148: END;
[000]  149:
[C68]  150: FUNCTION bogen(grad :REAL) :REAL;
[PH7]  151: BEGIN bogen:=grad*0.017453 END;
[000]  152:
[YTG]  153: PROCEDURE matr3d(x,y,z :INTEGER); VAR cx,cy
          ,cz,sx,sy,sz :REAL;
[B21]  154: BEGIN
[F5A]  155:   cz:=COS(bogen(z)); sz:=SIN(bogen(z));
[F4A]  156:   cx:=COS(bogen(x)); sx:=SIN(bogen(x));
[Z4A]  157:   cy:=COS(bogen(y)); sy:=SIN(bogen(y));
```

```
[S4F]  158:    am:=cy*cz-sz*sx*sy;  bm:=-cy*sz-sx*cz*sy;
           cm:=cx*sy;
[TMA]  159:    dm:=sz*cx;           em:=cx*cz;
           fm:=sx;
[D5F]  160:    gm:=-cz*sy-sz*sx*cy; hm:=sz*sy-sx*cz*cy;
           im:=cx*cy;
[SM9]  161:    x1:=-d1*cx*sz; y1:=-d1*cx*cz; z1:=-d1*sx
[TMO]  162: END;
[OOO]  163:
[S48]  164: PROCEDURE size; VAR d3 :INTEGER;
[64A]  165: BEGIN d3:=d+zo*10; d1:=d3/b; d2:=d1*g*b END
           ;
[OOO]  166:
[KYE]  167: PROCEDURE pro2d(x,y,z :INTEGER); VAR x2,y2,
           z2,x3,y3,z3 :REAL;
[B21]  168: BEGIN
[Z89]  169:    x3:=d2*x-x1; y3:=d2*y-y1; z3:=d2*z-z1;
[03H]  170:    x2:=am*x3+bm*y3+cm*z3; y2:=dm*x3+em*y3+fm
           *z3; z2:=gm*x3+hm*y3+im*z3;
[5ZC]  171:    u1:=ROUND(ox+26*d1*x2/y2); v1:=ROUND(oy-2
           0*d1*z2/y2);
[TMO]  172: END;
[OOO]  173:
[KV7]  174: PROCEDURE zeichne(i :INTEGER);
[B21]  175: BEGIN
[FEF]  176:    pro2d(p[ABS(e[i])][1],p[ABS(e[i])][2],p[A
           BS(e[i])][3]);
[V2F]  177:    IF (e[i]>0) THEN line(u2,v2,u1,v1,col[akt
           ]); u2:=u1; v2:=v1;
[TMO]  178: END;
[OOO]  179:
[WN7]  180: PROCEDURE axe(x,y,z :INTEGER);
[B21]  181: BEGIN
[9N7]  182:    pro2d(-x,-y,-z); u2:=u1; v2:=v1;
[VR9]  183:    pro2d(x,y,z); line(u2,v2,u1,v1,col[2])
[TMO]  184: END;
[OOO]  185:
[RMB]  186: FUNCTION inscr(spalte,zeile :BYTE) :BOOLEAN
           ;
[B21]  187: BEGIN
[UCG]  188:    IF (spalte>1) AND (spalte<79) AND (zeile>
           1) AND (zeile<25) THEN
[6X7]  189:       inscr:=TRUE ELSE inscr:=FALSE
[TMO]  190: END;
[OOO]  191:
[7OD]  192: PROCEDURE taxe(kt :CHAR); VAR spalte,zeile
           :BYTE;
[B21]  193: BEGIN
[8BB]  194:    spalte:=gxpos(u1); zeile:=gypos(v1);
[NFH]  195:    IF inscr(spalte,zeile) THEN text(kt,spalt
           e,zeile,col[10])
[TMO]  196: END;
[OOO]  197:
[H19]  198: PROCEDURE koorkreuz; CONST kg =800;
[B21]  199: BEGIN
[HRG]  200:    axe(kg,0,0); taxe('x'); axe(0,kg,0); taxe
           ('y'); axe(0,0,kg); taxe('z');
[TMO]  201: END;
[OOO]  202:
[AR5]  203: PROCEDURE getorigin;
[B21]  204: BEGIN
[GU1]  205:    CASE t OF
[F1B]  206:       'H' :BEGIN oy:=oy-25; IF oy<-175 THEN o
           y:=-175; END;
[OYA]  207:       'P' :BEGIN oy:=oy+25; IF oy>450 THEN oy
           :=450; END;
[9XA]  208:       'K' :BEGIN ox:=ox-30; IF ox<-100 THEN o
           x:=-100; END;
[5XA]  209:       'M' :BEGIN ox:=ox+30; IF ox>740 THEN ox
           :=740; END;
[TMO]  210:    END;
[TMO]  211: END;
[OOO]  212:
[3U4]  213: PROCEDURE getkey;
[B21]  214: BEGIN
[2C2]  215:    READ(KBD,t);
[TNF]  216:    If (t=#27) AND KEYPRESSED THEN BEGIN READ
           (KBD,t); getorigin END;
[TMO]  217: END;
[OOO]  218:
[KC8]  219: PROCEDURE listdata(i :INTEGER);
[B21]  220: BEGIN
[2VG]  221:    text('edge(',1,1,col[7]); wrt(i); txt(')(
           ',col[7]); wrt(e[i]);
[CMD]  222:    txt(') point(',col[7]); wrt(ABS(e[i])); t
           xt(')x(',col[7]);
[ZOA]  223:    wrt(p[ABS(e[i])][1]); txt(')y(',col[7]);
[61A]  224:    wrt(p[ABS(e[i])][2]); txt(')z(',col[7]);
[AAA]  225:    wrt(p[ABS(e[i])][3]); txt(')          ',co
           l[7]);
[P42]  226:    READ(KBD,t)
[TMO]  227: END;
[OOO]  228:
[ZGF]  229: PROCEDURE showdata(i :INTEGER); VAR spalte,
           zeile :BYTE;
```

```
[DD1]  230: BEGIN;
[8BB]  231:    spalte:=gxpos(u1); zeile:=gypos(v1);
[7EH]  232:    IF inscr(spalte,zeile) THEN wert(ABS(e[i]
           ),spalte,zeile)
[TMO]  233: END;
[OOO]  234:
[HFC]  235: FUNCTION getakt(n :INTEGER) :BYTE; VAR s :I
           NTEGER;
[CLF]  236: BEGIN s:=en DIV sn; IF n>s THEN getakt:=3 E
           LSE getakt:=11 END;
[OOO]  237:
[3YC]  238: PROCEDURE draw3d(bo :BOOLEAN); LABEL 0; VAR
            i :INTEGER;
[B21]  239: BEGIN
[F3A]  240: 0:clega; IF axs THEN koorkreuz;
[3GA]  241:    IF key THEN text(esc+okey,1,25,col[7]);
[MO3]  242:    akt:=11; i:=1;
[XG1]  243:    REPEAT
[BW7]  244:       IF shw THEN akt:=getakt(i);
[LXK]  245:       zeichne(i); IF bo THEN showdata(i); IF
           key THEN listdata(i); i:=SUCC(i);
[9Y4]  246:    UNTIL (i>en) OR (t=#27);
[6TD]  247:    IF NOT key THEN getkey; IF t IN ['H','P',
           'K','M'] THEN GOTO 0
[TMO]  248: END;
[OOO]  249:
[GWF]  250: PROCEDURE views; VAR lw :punkt; j :INTEGER;
           bo: BOOLEAN;
[B21]  251: BEGIN
[PE5]  252:    bo:=key; key:=FALSE;
[XG1]  253:    REPEAT
[ZC9]  254:       FOR j:=1 TO 3 DO lw[j]:=ROUND(RANDOM*36
           0);
[VFB]  255:       matr3d(lw[1],lw[2],lw[3]); draw3d(FALSE
           );
[OG4]  256:       IF t=#13 THEN w:=lw;
[767]  257:    UNTIL (t IN [#13,#27]); key:=bo
[TMO]  258: END;
[OOO]  259:
[SKG]  260: FUNCTION realein :REAL; VAR qs :STRING[8];
           q :REAL; ck :INTEGER;
[B21]  261: BEGIN
[AD9]  262:    ck:=1; cursor(TRUE); TEXTCOLOR(col[14]);
[FWB]  263:    READ(qs); cursor(FALSE); TEXTCOLOR(col[7]
           );
[NR6]  264:    IF qs<>'' THEN VAL(qs,q,ck);
[2VA]  265:    IF ck=0 THEN realein:=q ELSE realein:=0
[TMO]  266: END;
[OOO]  267:
[DO7]  268: FUNCTION intein :INTEGER;
[5K8]  269: BEGIN intein:=ROUND(realein) END;
[OOO]  270:
[ON7]  271: FUNCTION winkelein :INTEGER;
[GTA]  272: BEGIN winkelein:=ABS(intein) MOD 360 END;
[OOO]  273:
[HY4]  274: PROCEDURE degrees;
[B21]  275: BEGIN
[2BC]  276:    WRITE('Enter degrees  WX=:'); w[1]:=winke
           lein;
[DCG]  277:    WRITE(' WY=:'); w[2]:=winkelein; WRITE('
           WZ=:'); w[3]:=winkelein
[TMO]  278: END;
[OOO]  279:
[2T3]  280: PROCEDURE dir;
[SRH]  281: VAR inhalt :TEXT; ebuff :STRING[80]; abuff
           :STRING[8]; n,m :INTEGER;
[B21]  282: BEGIN
[WF2]  283:    GOTOXY(1,3);
[U3D]  284:    ASSIGN(inhalt,dtyp); {$I-} RESET(inhalt);
           {$I+}
[RE5]  285:    IF IORESULT=0 THEN BEGIN
[926]  286:       TEXTCOLOR(col[6]); n:=0;
[UB9]  287:       FOR m:=1 TO 4 DO READLN(inhalt,ebuff);
[G37]  288:       WHILE NOT EOF(inhalt) DO BEGIN
[PRB]  289:          READLN(inhalt,ebuff); abuff:=COPY(ebu
           ff,1,8);
[4TA]  290:          IF (abuff[1]<>'.') AND (abuff[1]<>' '
           ) THEN BEGIN
[OLA]  291:             n:=n+1; IF n>7 THEN BEGIN WRITE('
           '); n:=1; END;
[5E3]  292:             WRITE(abuff,'  ')
[TMO]  293:          END;
[TMO]  294:       END;
[9C8]  295:       CLOSE(inhalt); TEXTCOLOR(col[7])
[3B3]  296:    END ELSE BEGIN
[V6H]  297:       txt(dtyp,col[12]); txt(' not found',col
           [4]); TEXTCOLOR(col[7])
[TMO]  298:    END;
[C82]  299:    GOTOXY(1,1)
[TMO]  300: END;
```

```
[000]  301:
[KM8]  302: PROCEDURE namup; VAR n :INTEGER;
[WPE]  303: BEGIN FOR n:=1 TO LENGTH(nam) DO nam[n]:=UP
       CASE(nam[n]) END;
[000]  304:
[XS4]  305: PROCEDURE namein;
[B21]  306: BEGIN
[968]  307:   nam1:=nam; WRITE(' filename: ');
[XL8]  308:   cursor(TRUE); TEXTCOLOR(col[14]);
[5XB]  309:   READ(nam); cursor(FALSE); TEXTCOLOR(col[7
       ]);
[CT3]  310:   namup; WRITE('    ')
[TMO]  311: END;
[000]  312:
[CK8]  313: PROCEDURE save; VAR i,j :INTEGER;
[B21]  314: BEGIN
[LD5]  315:   IF (nam>'') THEN BEGIN
[42B]  316:     ASSIGN(daf,nam+'.3D'); {$I-} REWRITE(da
       f); {$I+}
[RE5]  317:     IF IORESULT=0 THEN BEGIN
[375]  318:       WRITE(daf,pn,en,sn);
[N7D]  319:       FOR i:=1 TO pn DO FOR j:=1 TO 3 DO WR
       ITE(daf,p[i][j]);
[NE8]  320:       FOR i:=1 TO en DO WRITE(daf,e[i]);
[2C2]  321:       CLOSE(daf);
[VE6]  322:       pn1:=pn; en1:=en; sn1:=sn
[DDO]  323:     END
[YY1]  324:     ELSE BEGIN
[3K4]  325:       TEXTCOLOR(col[4]);
[P5G]  326:       WRITE('Disk missing or invalid filena
       me press any key');
[Z48]  327:       REPEAT UNTIL KEYPRESSED; nam:=nam1
[TMO]  328:     END;
[7E4]  329:   END ELSE nam:=nam1
[TMO]  330: END;
[000]  331:
[OG8]  332: PROCEDURE load; VAR i,j :INTEGER;
[B21]  333: BEGIN
[LD5]  334:   IF (nam>'') THEN BEGIN
[745]  335:     ASSIGN(daf,nam+'.3D');
[MC6]  336:     {$I-} RESET(daf); {$I+}
[RE5]  337:     IF IORESULT=0 THEN BEGIN
[EN4]  338:       READ(daf,pn,en,sn);
[KNC]  339:       FOR i:=1 TO pn DO FOR j:=1 TO 3 DO R
       EAD(daf,p[i][j]);
[6Z7]  340:       FOR i:=1 TO en DO READ(daf,e[i]);
[2C2]  341:       CLOSE(daf);
[HK6]  342:       pn1:=pn; en1:=en; sn1:=sn;
[DDO]  343:     END
[YY1]  344:     ELSE BEGIN
[Y7H]  345:       txt(nam+'.3D',col[12]); txt(' not fou
       nd press any key',col[4]);
[TGB]  346:       REPEAT UNTIL KEYPRESSED; nam:=nam1; I
       Oer:=TRUE
[TMO]  347:     END;
[7E4]  348:   END ELSE nam:=nam1
[TMO]  349: END;
[000]  350:
[ZGB]  351: PROCEDURE changedir; VAR direc :STRING[20];
[B21]  352: BEGIN
[TBA]  353:   GETDIR(0,direc); text(direc,1,3,col[6]);
[8RA]  354:   text('Change directory to: ',1,1,col[7]);
[XL8]  355:   cursor(TRUE); TEXTCOLOR(col[14]);
[OPB]  356:   READ(direc); cursor(FALSE); TEXTCOLOR(col
       [7]);
[D76]  357:   {$I-} CHDIR(direc); {$I+}
[KCL]  358:   IF (IORESULT<>0) THEN BEGIN txt('  Invali
       d path press any key',col[4]);
[BL5]  359:     REPEAT UNTIL KEYPRESSED
[TMO]  360:   END;
[TMO]  361: END;
[000]  362:
[SR4]  363: PROCEDURE merge;
[KW6]  364: VAR i,lpn,len,lsn :INTEGER;
[KKE]  365:     lp :ARRAY[1..max] OF punkt; le :ARRAY[1
       ..max] OF INTEGER;
[B21]  366: BEGIN
[Z87]  367:   lpn:=pn; len:=en; lsn:=sn;
[758]  368:   FOR i:=1 TO pn DO lp[i]:=p[i];
[E18]  369:   FOR i:=1 TO en DO le[i]:=e[i];
[W86]  370:   load; IF NOT IOer THEN BEGIN
[YHA]  371:     IF (pn+lpn>max)OR(en+len>max) THEN BEGI
       N
[Y6G]  372:       txt('   '+memo,col[4]); REPEAT UNTIL
       KEYPRESSED; TEXTCOLOR(col[7]);
[SH7]  373:       pn:=lpn; en:=len; sn:=lsn;
[RW7]  374:       FOR i:=1 TO pn DO p[i]:=lp[i];
[9T7]  375:       FOR i:=1 TO en DO e[i]:=le[i];
[ZG6]  376:       loeschfeld; nam:=nam1
[3B3]  377:     END ELSE BEGIN
[RW7]  378:       IF (len=en) AND (lpn=pn) THEN
[10C]  379:         IF (lsn=sn) THEN sn:=sn*2 ELSE sn:=
       2 ELSE sn:=1;
```

```
[HSH]  380:       FOR i:=1 TO len DO IF (le[i]<0) THEN
       eck(le[i]-pn) ELSE eck(le[i]+pn);
[16E]  381:       FOR i:=1 TO lpn DO pkt(lp[i][1],lp[i]
       [2],lp[i][3]);
[8KF]  382:       IF LENGTH(nam)=8 THEN DELETE(nam,8,1)
       ; INSERT('*',nam,LENGTH(nam)+1)
[TMO]  383:     END;
[TMO]  384:   END;
[TMO]  385: END;
[000]  386:
[9W9]  387: PROCEDURE exchange(pan,pen :INTEGER);
[X5B]  388: VAR lp :ARRAY[1..max] OF punkt; i :INTEGER;
[B21]  389: BEGIN
[KM7]  390:   txt('Exchange axis  ',col[7]);
[MU8]  391:   txt('1',col[15]); txt('(xy) ',col[7]);
[WU8]  392:   txt('2',col[15]); txt('(xz) ',col[7]);
[3V8]  393:   txt('3',col[15]); txt('(yz) ',col[7]);
[6OE]  394:   txt('O',col[15]); txt('(xyz)'+okey,col[7]
       ); READ(KBD,t);
[932]  395:   WRITELN;
[OV5]  396:   FOR i:=pan TO pen DO BEGIN
[9O3]  397:     lp[i]:=p[i];
[GU1]  398:     CASE t OF
[GBD]  399:       '1' :BEGIN p[i][1]:=lp[i][2]; p[i][2]
       :=lp[i][1] END;
[TBD]  400:       '2' :BEGIN p[i][1]:=lp[i][3]; p[i][3]
       :=lp[i][1] END;
[2CD]  401:       '3' :BEGIN p[i][2]:=lp[i][3]; p[i][3]
       :=lp[i][2] END;
[8CL]  402:       'O' :BEGIN p[i][1]:=lp[i][2]; p[i][2]
       :=lp[i][3]; p[i][3]:=lp[i][1] END;
[TMO]  403:     END;
[TMO]  404:   END;
[TMO]  405: END;
[000]  406:
[E5G]  407: PROCEDURE fractal(pan,pen :INTEGER); VAR i,
       j,n,ck,rf :INTEGER;
[B21]  408: BEGIN
[5XA]  409:   txt('Fractal deformation of  ',col[7]);
[H69]  410:   txt('O',col[15]); txt('(xyz) ',col[7]);
[DN8]  411:   txt('1',col[15]); txt('(x) ',col[7]);
[MN8]  412:   txt('2',col[15]); txt('(y) ',col[7]);
[OED]  413:   txt('3',col[15]); txt('(z)'+okey,col[7]);
       READ(KBD,t);
[932]  414:   WRITELN;
[136]  415:   IF t IN ['O'..'3'] THEN BEGIN
[DGD]  416:     WRITE('Enter random factor: '); rf:=int
       ein; WRITELN;
[2G4]  417:     IF rf<0 THEN rf:=0;
[AU2]  418:     VAL(t,n,ck);
[CL4]  419:     FOR i:=pan TO pen DO
[U2C]  420:     IF (n>0) THEN p[i][n]:=p[i][n]+ROUND(RA
       NDOM*rf)
[CND]  421:               ELSE FOR j:=1 TO 3 DO p[i][j]:
       =p[i][j]+ROUND(RANDOM*rf);
[TMO]  422:   END;
[TMO]  423: END;
[000]  424:
[KN8]  425: PROCEDURE resetfh; VAR j :INTEGER;
[6DB]  426: BEGIN FOR j:=1 TO 3 DO BEGIN f[j]:=1; h[j]:
       =0 END END;
[000]  427:
[195]  428: PROCEDURE einplace;
[B21]  429: BEGIN
[4TA]  430:   WRITE('Enter dislocation Distances');
[TC7]  431:   WRITE('   X+:'); h[1]:=realein;
[OD7]  432:   WRITE('   Y+:'); h[2]:=realein;
[GG9]  433:   WRITE('   Z+:'); h[3]:=realein; WRITELN
[TMO]  434: END;
[000]  435:
[AL8]  436: PROCEDURE einform; VAR j :INTEGER;
[B21]  437: BEGIN
[6F8]  438:   WRITE('Enter deform faktor        ');
[HC7]  439:   WRITE('   X*:'); f[1]:=realein;
[SC7]  440:   WRITE('   Y*:'); f[2]:=realein;
[8G9]  441:   WRITE('   Z*:'); f[3]:=realein; WRITELN;
[RB9]  442:   FOR j:=1 TO 3 DO IF f[j]=0 THEN f[j]:=1
[TMO]  443: END;
[000]  444:
[XV9]  445: PROCEDURE placeform; VAR i,j :INTEGER;
[B21]  446: BEGIN
[MYK]  447:   FOR i:=1 to pn DO FOR j:=1 TO 3 DO p[i][j
       ]:=ROUND((p[i][j]+h[j])*f[j]);
[9X2]  448:   resetfh;
[TMO]  449: END;
[000]  450:
[2X4]  451: PROCEDURE entto;
[B21]  452: BEGIN
[BUA]  453:   txt('Press ',col[7]); txt('ENTER',col[15]
       );
[CL4]  454:   txt(' to ',col[7])
[TMO]  455: END;
```

```
[000]  456:
[HCG]  457: PROCEDURE addiere; VAR o :punkt; i,j,spn,se
           n,lpn :INTEGER;
[B21]  458: BEGIN
[T1C]  459:   sen:=en DIV sn; spn:=pn DIV sn; lpn:=pn;
[54C]  460:   IF ((pn+spn>max) OR (en+sen>max)) THEN BE
           GIN
[9ZA]  461:     txt(memo,col[4]); REPEAT UNTIL KEYPRESS
           ED
[DDO]  462:   END
[YY1]  463:   ELSE BEGIN
[WS5]  464:     einplace; einform;
[G66]  465:     FOR i:=1 TO spn DO BEGIN
[O8C]  466:       FOR j:=1 TO 3 DO o[j]:=ROUND((p[i][j]
           +h[j])*f[j]);
[LS4]  467:       pkt(o[1],o[2],o[3])
[TMO]  468:     END;
[WBH]  469:     FOR i:=1 TO sen DO IF e[i]<0 THEN eck(e
           [i]-lpn) ELSE eck(e[i]+lpn);
[SSG]  470:     resetfh; exchange(lpn+1,pn); fractal(lp
           n+1,pn); sn:=sn+1;
[BE4]  471:     IF (sn=2) THEN BEGIN
[9XE]  472:       entto; txt('to link ',col[7]); wrt(en
           ); txt(' edges',col[7]);
[2C2]  473:       READ(KBD,t);
[4H4]  474:       IF (t=#13) THEN BEGIN
[5S3]  475:         lpn:=pn DIV 2;
[KGC]  476:         FOR i:=1 TO lpn DO BEGIN eck(-i); e
           ck(i+lpn) END;
[8A1]  477:         sn:=1;
[TMO]  478:       END;
[TMO]  479:     END;
[TMO]  480:   END;
[TMO]  481: END;
[000]  482:
[G28]  483: PROCEDURE kill; VAR i :INTEGER;
[B21]  484: BEGIN
[WH9]  485:   WRITE('Actuell shapes '); TEXTCOLOR(14);
[1PE]  486:   WRITE(sn); TEXTCOLOR(7); WRITE(' enter sh
           apes to kill: ');
[R16]  487:   i:=ABS(intein); WRITELN;
[SKG]  488:   IF i<=sn THEN BEGIN pn:=pn-i*(pn DIV sn);
           en:=en-i*(en DIV sn) END;
[Z72]  489:   sn:=sn-i
[TMO]  490: END;
[000]  491:
[X8C]  492: PROCEDURE polyeder; VAR n,x,y,z,lpn :INTEGE
           R;
[DN5]  493:   PROCEDURE tetraeder;
[B21]  494:   BEGIN
[RTA]  495:     pkt(n+x,n+y,n+z);   pkt(n+x,-n+y,-n+z);
[AHA]  496:     pkt(-n+x,n+y,-n+z); pkt(-n+x,-n+y,n+z);
[FGA]  497:     eck(-pn+3); eck(pn-2); eck(pn-1); eck(p
           n);
[4TA]  498:     eck(-pn+2); eck(pn); eck(pn-3); eck(pn-
           1);
[TMO]  499:   END;
[O25]  500:   PROCEDURE wuerfel;
[B21]  501:   BEGIN
[N69]  502:     pkt(-n+x,-n+y,-n+z);   pkt(-n+x,n+y,-n+
           z);
[GB9]  503:     pkt(n+x,n+y,-n+z);     pkt(n+x,-n+y,-n+
           z);
[BW9]  504:     pkt(-n+x,-n+y,n+z);    pkt(-n+x,n+y,n+z
           );
[O5A]  505:     pkt(n+x,n+y,n+z);      pkt(n+x,-n+y,n+z
           );
[VZD]  506:     eck(-pn+7); eck(pn-6); eck(pn-5); eck(p
           n-4); eck(pn-7);
[S5D]  507:     eck(-pn+3); eck(pn-2); eck(pn-1); eck(p
           n); eck(pn-3);
[AYA]  508:     eck(-pn+7); eck(pn-3); eck(-pn+6); eck(
           pn-2);
[8GA]  509:     eck(-pn+5); eck(pn-1); eck(-pn+4); eck(
           pn);
[TMO]  510:   END;
[P75]  511:   PROCEDURE oktaeder;
[B21]  512:   BEGIN
[867]  513:     pkt(x,-n+y,z);   pkt(n+x,y,z);
[RO8]  514:     pkt(x,n+y,z);    pkt(-n+x,y,z);
[WY7]  515:     pkt(x,y,n+z);    pkt(x,y,-n+z);
[OWA]  516:     eck(-pn+5); eck(pn-4); eck(pn-3); eck(p
           n-2);
[99A]  517:     eck(pn-5); eck(pn-1); eck(pn-3); eck(pn
           );
[L1D]  518:     eck(pn-4); eck(pn-1); eck(pn-2); eck(pn)
           ; eck(pn-5);
[TMO]  519:   END;
[B21]  520: BEGIN
[4LB]  521:   txt('T',col[15]); txt('etraeder  ',col[7]
           );
[UBA]  522:   txt('W',col[15]); txt('uerfel  ',col[7]);
[6YE]  523:   txt('O',col[15]); txt('ktaeder  '+okey,co
           l[7]); READ(KBD,t);
```

```
[SW2]  524:   t:=UPCASE(t);
[4O7]  525:   IF t IN ['O','T','W'] THEN BEGIN
[CZM]  526:     WRITELN; lpn:=pn; WRITE('Random polyede
           r or enter size: '); n:=intein;
[WN3]  527:     IF n<>0 THEN BEGIN
[HO6]  528:       WRITE('  X=:'); x:=intein;
[FS5]  529:       WRITE(' Y=:'); y:=intein;
[9X7]  530:       WRITE(' Z=:'); z:=intein; WRITELN;
[3B3]  531:     END ELSE BEGIN
[ZF7]  532:       WRITELN; n:=ROUND(RANDOM*180)+20;
[6M5]  533:       x:=ROUND(RANDOM*1200)-600;
[7M5]  534:       y:=ROUND(RANDOM*1200)-600;
[8M5]  535:       z:=ROUND(RANDOM*1200)-600;
[TMO]  536:     END;
[66F]  537:     CASE t OF 'T':tetraeder; 'W':wuerfel; '
           O':oktaeder; END;
[SXB]  538:     exchange(lpn+1,pn); fractal(lpn+1,pn);
           sn:=1
[TMO]  539:   END;
[TMO]  540: END;
[000]  541:
[UG6]  542: FUNCTION chksn :INTEGER;
[LRD]  543: BEGIN IF (pn=0) AND (en=0) THEN chksn:=0 EL
           SE chksn:=1 END;
[000]  544:
[M6E]  545: PROCEDURE listpointedge(bo :BOOLEAN; evn :I
           NTEGER);
[B21]  546: BEGIN
[4W8]  547:   GOTOXY(1,2); WRITE(spc); GOTOXY(1,2);
[92G]  548:   IF bo THEN BEGIN txt('edge ',col[7]); wrt
           (evn); txt(' (',col[7]);
[4M7]  549:     wrt(e[evn]); txt(')=:',col[7])
[3B3]  550:   END ELSE BEGIN
[6KC]  551:     txt('point ',col[7]); wrt(evn); txt(' (
           ',col[7]);
[64H]  552:     TEXTCOLOR(col[14]); WRITE(p[evn][1],' ',
           p[evn][2],' ',p[evn][3]);
[5U3]  553:     txt(')  ',col[7])
[TMO]  554:   END;
[TMO]  555: END;
[000]  556:
[725]  557: PROCEDURE entveri;
[B21]  558: BEGIN
[ZLA]  559:   txt(' press ',col[7]); txt('ENTER ',col[1
           5]);
[M27]  560:   txt('to verify',col[7])
[TMO]  561: END;
[000]  562:
[DHC]  563: PROCEDURE points; VAR o :punkt; vin :INTEGE
           R;
[B21]  564: BEGIN
[H39]  565:   text('Actuell points ',1,1,col[7]);
[RGC]  566:   wrt(pn); txt(' enter point to start: ',co
           l[7]);
[158]  567:   vin:=intein; IF vin>0 THEN BEGIN
[XG1]  568:     REPEAT
[VU7]  569:       listpointedge(FALSE,vin);
[X36]  570:       WRITE(' X=:'); o[1]:=intein;
[O46]  571:       WRITE(' Y=:'); o[2]:=intein;
[346]  572:       WRITE(' Z=:'); o[3]:=intein;
[9AC]  573:       WRITE('  '); entveri; READ(KBD,t); IF
           t=#13 THEN BEGIN
[Z76]  574:         p[vin]:=o; vin:=vin+1;
[TMO]  575:       END;
[3W2]  576:     UNTIL t<>#13;
[8U8]  577:     vin:=vin-1; IF (pn<>vin) THEN BEGIN
[SU9]  578:       GOTOXY(1,3); TEXTCOLOR(col[14]); WRIT
           E(vin);
[5UA]  579:       txt(' points',col[7]); entveri; READ(
           KBD,t);
[RNA]  580:       IF t=#13 THEN BEGIN pn:=vin; sn:=chks
           n END;
[GO5]  581:     END; loeschfeld
[TMO]  582:   END;
[TMO]  583: END;
[000]  584:
[979]  585: PROCEDURE edges; VAR i,vin :INTEGER;
[B21]  586: BEGIN
[3E8]  587:   text('Actuell edges ',1,1,col[7]);
[94C]  588:   wrt(en); txt(' enter edge to start: ',col
           [7]);
[158]  589:   vin:=intein; IF vin>0 THEN BEGIN
[XG1]  590:     REPEAT
[XMA]  591:       listpointedge(TRUE,vin); i:=intein;
[XAB]  592:       IF i<>0 THEN BEGIN e[vin]:=i; vin:=vi
           n+1 END;
[YP5]  593:     UNTIL i=0; vin:=vin-1;
[BB8]  594:     IF (en<>vin) THEN BEGIN GOTOXY(1,3);
[SAG]  595:       TEXTCOLOR(col[14]); WRITE(vin); txt('
           edges',col[7]); entveri;
[2C2]  596:       READ(KBD,t);
[OMA]  597:       IF t=#13 THEN BEGIN en:=vin; sn:=chks
```

```
          n END;
[G05]  598:     END; loeschfeld
[TMO]  599:   END;
[TMO]  600: END;
[000]  601:
[YN8]  602: PROCEDURE onedge; VAR i :INTEGER;
[B21]  603: BEGIN
[MMC]  604:   entto; WRITE('link '); TEXTCOLOR(col[14])
          ; WRITE(pn);
[LX7]  605:   txt(' points',col[7]); READ(KBD,t);
[ZPE]  606:   IF t=#13 THEN BEGIN en:=0; eck(-1); FOR i
          :=2 TO pn DO eck(i); END;
[TMO]  607: END;
[000]  608:
[VOB]  609: PROCEDURE uneck(n :INTEGER); VAR i :INTEGER
          ;
[Z3G]  610: BEGIN en:=en-1; ek:=ek+1; FOR i:=n TO en DO
          e[i]:=e[SUCC(i)] END;
[000]  611:
[TBC]  612: PROCEDURE unpkt(n :INTEGER); VAR i,ck :INTE
          GER;
[B21]  613: BEGIN
[2U4]  614:   pn:=pn-1; pk:=pk+1;
[3B8]  615:   FOR i:=n TO pn DO p[i]:=p[SUCC(i)];
[2GG]  616:   FOR i:=1 TO en DO IF (ABS(e[i])=n) THEN e
          [SUCC(i)]:=-1*ABS(e[SUCC(i)]);
[XG1]  617:   REPEAT
[D91]  618:     ck:=0;
[3VA]  619:     FOR i:=1 TO en DO IF (ABS(e[i])=n) THEN
           ck:=i;
[T66]  620:     IF (ck>0) THEN uneck(ck);
[5D2]  621:   UNTIL (ck=0);
[4BA]  622:   FOR i:=1 TO en DO IF (ABS(e[i])>n) THEN B
          EGIN
[V6E]  623:     IF (e[i]<0) THEN e[i]:=SUCC(e[i]) ELSE
          e[i]:=PRED(e[i]);
[TMO]  624:   END;
[TMO]  625: END;
[000]  626:
[4BA]  627: PROCEDURE optimum; VAR i,j,ck :INTEGER;
[B21]  628: BEGIN
[T6D]  629:   entto; WRITE('get optimum shape'); READ(K
          BD,t);
[4H4]  630:   IF (t=#13) THEN BEGIN
[U58]  631:     txt(' ... please wait',col[4]);
[GR5]  632:     FOR j:=1 TO pn DO BEGIN
[XG1]  633:       REPEAT
[AB5]  634:         ck:=0; FOR i:=1 TO pn-1 DO
[Y77]  635:         IF (i<>j) AND (p[j][1]=p[i][1])
[3B5]  636:                   AND (p[j][2]=p[i][2])
[V48]  637:                   AND (p[j][3]=p[i][3]) THE
          N ck:=i;
[GB4]  638:         IF (ck>0) THEN BEGIN
[WNA]  639:           FOR i:=1 TO en DO IF (ABS(e[i])=c
          k) THEN BEGIN
[LD9]  640:             IF (e[i]<0) THEN e[i]:=-j ELSE
          e[i]:=j
[TMO]  641:           END;
[NP2]  642:           unpkt(ck);
[TMO]  643:         END;
[5D2]  644:       UNTIL (ck=0);
[TMO]  645:     END;
[XG1]  646:     REPEAT
[6ZE]  647:       ck:=0; FOR i:=1 TO en DO IF (e[i]<0)
          AND (e[SUCC(i)]<0) THEN ck:=i;
[T66]  648:       IF (ck>0) THEN uneck(ck);
[5D2]  649:     UNTIL (ck=0);
[XG1]  650:     REPEAT
[5SE]  651:       ck:=0; FOR i:=1 TO en DO IF (e[PRED(i
          )]=ABS(e[i])) THEN ck:=i;
[T66]  652:       IF (ck>0) THEN uneck(ck);
[5D2]  653:     UNTIL (ck=0);
[TMO]  654:   END;
[TMO]  655: END;
[000]  656:
[U49]  657: PROCEDURE delins; VAR vin :INTEGER;
[B21]  658: BEGIN
[CC2]  659:   ek:=0; pk:=0;
[25B]  660:   text('Kill edge: ',1,1,col[7]); vin:=inte
          in;
[HGB]  661:   IF (vin>0) THEN BEGIN uneck(vin); sn:=1 E
          ND;
[E3A]  662:   txt('   point: ',col[7]); vin:=intein;
[7KB]  663:   IF (vin>0) THEN BEGIN unpkt(vin); sn:=1 E
          ND;
[767]  664:   WRITELN; WRITELN; optimum;
[EZC]  665:   GOTOXY(1,14); wrt(ek); txt(' edges ',col[
          7]); wrt(pk);
[1FC]  666:   txt(' points killed press any key',col[7]
          );
[BL5]  667:   REPEAT UNTIL KEYPRESSED
[TMO]  668: END;
```

```
[000]  669:
[K78]  670: FUNCTION bochk(bo :BOOLEAN) :CHAR;
[W3C]  671: BEGIN IF bo=TRUE THEN bochk:='+' ELSE bochk
          :='-' END;
[000]  672:
[N56]  673: PROCEDURE turnaround;
[E3D]  674: VAR px,pz :ARRAY[1..max] OF INTEGER; i,grad
          :INTEGER;
[B21]  675: BEGIN
[MVA]  676:   WRITE('Enter degrees to turn shape: ');
[TLA]  677:   grad:=winkelein; IF grad=0 THEN EXIT;
[6W8]  678:   txt('   ... please wait',col[4]);
[XLE]  679:   FOR i:=1 TO pn DO BEGIN px[i]:=p[i][1]; p
          z[i]:=p[i][3] END;
[BR5]  680:   FOR i:=1 TO pn DO BEGIN
[GWK]  681:     p[i][1]:=ROUND(SIN(bogen(grad))*px[i])+
          ROUND(SIN(bogen(grad+90))*pz[i]);
[PVK]  682:     p[i][3]:=ROUND(COS(bogen(grad))*px[i])+
          ROUND(COS(bogen(grad+90))*pz[i]);
[TMO]  683:   END;
[TMO]  684: END;
[000]  685:
[UB5]  686: PROCEDURE rotation;
[MPF]  687: VAR lp :ARRAY[1..max] OF punkt; le :ARRAY[1
          ..max] OF INTEGER;
[VBE]  688:     x,y,z,lpn,len,i,j,wn,wn1 :INTEGER;  gra
          d,part :REAL;
[B21]  689: BEGIN
[6A8]  690:   WRITE('Enter number of lines:  ');
[7PC]  691:   j:=ABS(intein); WRITELN; IF (j=0) OR (en=
          0) THEN EXIT;
[489]  692:   IF (j>18) OR (max<en*j+pn*j) THEN BEGIN
[4WB]  693:     txt(memo,col[4]); REPEAT UNTIL KEYPRESS
          ED END
[YY1]  694:   ELSE BEGIN
[HM6]  695:     wn:=j; lpn:=pn; len:=en;
[KKD]  696:     WRITE('Enter part of a circle: '); part
          :=realein;
[PEB]  697:     IF (part<=1) THEN BEGIN part:=1; wn1:=w
          n-1 END
[7ND]  698:     ELSE wn1:=wn; txt(' ... please wait',co
          l[4]); WRITELN;
[C69]  699:     FOR i:=1 TO lpn DO lp[i]:=p[i];
[K19]  700:     FOR i:=1 TO len DO le[i]:=e[i];
[NC2]  701:     en:=0; pn:=0;
[KT5]  702:     FOR j:=0 TO wn1 DO BEGIN
[E66]  703:       grad:=j*(360/(wn*part));
[D56]  704:       FOR i:=1 TO lpn DO BEGIN
[RSL]  705:         x:=ROUND(SIN(bogen(grad))*lp[i][1])
          +ROUND(SIN(bogen(grad+90))*lp[i][3]);
[7A3]  706:         y:=lp[i][2];
[OTL]  707:         z:=ROUND(COS(bogen(grad))*lp[i][1])
          +ROUND(COS(bogen(grad+90))*lp[i][3]);
[XN2]  708:         pkt(x,y,z)
[TMO]  709:       END;
[1G4]  710:       FOR i:=1 TO len DO
[XPF]  711:       IF (le[i]<0) THEN eck(le[i]-lpn*j) EL
          SE eck(le[i]+lpn*j);
[TMO]  712:     END;
[X5A]  713:     entto; WRITE('link edges'); READ(KBD,t)
          ;
[AR3]  714:     IF t=#13 THEN BEGIN
[A78]  715:       FOR i:=1 TO lpn DO FOR j:=0 TO wn DO
[FT7]  716:       IF j=0 THEN eck(-ABS(le[i])) ELSE
[KSH]  717:       IF (j=wn) AND (part=1) THEN eck(ABS(l
          e[i])) ELSE eck(ABS(le[i])+lpn*j);
[6Z0]  718:       sn:=1
[NK4]  719:     END ELSE sn:=wn1+1
[TMO]  720:   END;
[TMO]  721: END;
[000]  722:
[976]  723: PROCEDURE funktionen;
[PKC]  724: VAR py :ARRAY[1..max] OF INTEGER; i,n,ck  :
          INTEGER;
[UC8]  725:     q  :REAL;                          sinus   :
          BOOLEAN;
[748]  726:   FUNCTION qxz(x,z :INTEGER) :REAL;
[799]  727:   BEGIN qxz:=SQRT(SQR(x/100)+SQR(z/100)) EN
          D;
[B21]  728: BEGIN
[MGA]  729:   text('1',1,1,col[15]); txt(' y=q(xz)',col
          [7]);
[1KC]  730:   text('2',1,2,col[15]); txt(' y=SQR(q(xz))
          ',col[7]);
[9CC]  731:   text('3',1,3,col[15]); txt(' y=SQRT(q(xz)
          )',col[7]);
[U9A]  732:   text('4',1,4,col[15]); txt(' y=x+z',col[7
          ]);
[HSB]  733:   text('5',1,5,col[15]); txt(' y=SQR(x+z)',
          col[7]);
[U1C]  734:   text('6',1,6,col[15]); txt(' y=SQRT(x+z)'
          ,col[7]);
[XAA]  735:   text('7',1,7,col[15]); txt(' y=x-z',col[7
          ]);
```

Raffinierte Lösungen zu Open Access, Lotus 1-2-3, Symphony und Framework

Dieses neue Nachschlagewerk bietet Ihnen

- **Kurse für effizientes Softwareengineering;**
- **Hilfestellung bei Auswahl und Anwendung von Softwarepaketen:** Ausführlich kommentierte Befehlsübersichten mit Beispielen für Dateiverwaltung, Textverarbeitung, Businessgrafik und Kalkulation. Schnellkurse machen Sie mit den integrierten Kommandosprachen wie z. B. FRED für Framework vertraut;
- **Musterlösungen, u. a. für technisch-mathematischen und betriebswirtschaftlichen Einsatz;** z. B. statistische Auswertungsverfahren, Unternehmens-, Marketing- und Vertriebsanalyse und finanzwirtschaftliche Templates, in die Sie lediglich noch Ihre Daten eintragen;
- **Add-on-Programme für Ihre Adreß- oder Artikelverwaltung, Fakturierung…** mit detaillierter Problembeschreibung, Bedienungsanleitung und Hinweisen zum Einbinden in Ihr Softwarepaket;
- **Utilities,** u. a. Routinen zur Diskettenverwaltung oder einen Menügenerator, mit dem Sie Programme über Funktionstasten aufrufen können;
- **Praxisnahe Konfigurationsbeispiele für die erfolgreiche PC-Vernetzung;**
- einen eigens eingerichteten **Diskettenservice;**
- **Ergänzungsausgaben zum Grundwerk** mit neuen Musterprogrammen, Templates, Tips und aktuellen Software-Informationen.

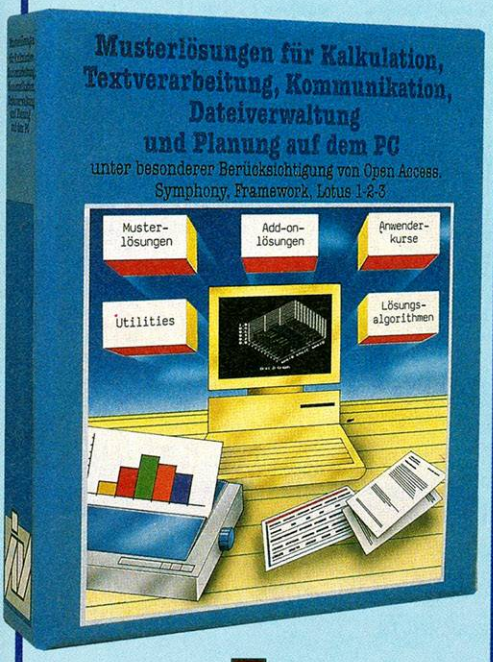
Fordern Sie noch heute an:

Musterlösungen für Kalkulation, Textverarbeitung, Kommunikation, Dateiverwaltung und Planung auf dem PC

unter besonderer Berücksichtigung von Open Access, Symphony, Framework, Lotus 1-2-3

stabiler Kunstlederordner im Format DIN A4, Grundwerk ca. 400 Seiten, Bestell Nr. 3300, Preis DM 92,–

Alle 2–3 Monate erhalten Sie Ergänzungsausgaben zum Grundwerk mit je ca. 120 Seiten zum Seitenpreis von 38 Pfennig (Abbestellung jederzeit möglich).



INTEREST-VERLAG
Fachverlag für anspruchsvolle Freizeitgestaltung

Industriestraße 21
D-8901 Kissing
Tel. 08233/2101-0

Mehr Erfolg durch PC-Tuning

Dieses Nachschlagewerk bietet Ihnen

- **detailliertes Hardwarewissen** über PCs, XTs, ATs bis hin zu den neuen PS/2-Systemen. Umfassende Prozessorbeschreibungen der Intelserie 80(x)xx zeigen Ihnen die CPU-Architekturen und -Beschaltungen sowie die internen Abläufe bei Adressierung und Befehlsausführung;
- **Anleitungen für den routinierten Einsatz von Systemsoftware:** Sie lernen Aufbau und Funktionen von MS-/PC-DOS und Unix kennen und erfahren, welche Möglichkeiten Ihnen die System- und Benutzerschnittstellen bieten;
- **Tuning durch Hardwareerweiterungen,** u. a. Anleitung für den Festplatteneinbau; Einsatz von Coprozessoren sowie Bauanleitungen, z. B. für eine parallele I/O-Karte oder einen universellen EPROMer;
- **neue Tools und Utilities:** u. a. MS-DOS-Utilities, Zweifach-Tastaturtreiber, Mausanschluß für Turbo-Pascal, RAM-Disk, Turbo-Pascal-Schnittstelle zu MS-DOS, V.24-Treiber, Profiler;
- **den kompletten Assembler SIGMASM** für hardwarenahes Programmieren;
- **Softwareengineering:** u. a. mit ausführlich dokumentierten Programm-Strukturplänen für komplexe Aufgabenstellungen;
- **Musterlösungen,** z. B. zur Lagerverwaltung;
- **Know-how für spezielle PC-Bereiche** wie DFÜ oder Netzwerke;
- **Ergänzungsausgaben zum Grundwerk** mit neuer Anwendersoftware, weiteren Programmiersprachkursen wie C oder Lisp, aktuellen Hardware- und Betriebssystembeschreibungen (u. a. PCs mit 80386-Prozessor) sowie Tips und Utilities;

Zusätzlicher Diskettenservice!

Fordern Sie noch heute an:

Hard- und Softwaretuning für IBM PCs und Kompatible

stabiler Kunstlederordner, DIN A4, Grundwerk ca. 870 Seiten, Bestell-Nr. 2600, Preis DM 92,–

Alle 2–3 Monate erhalten Sie Ergänzungsausgaben zum Grundwerk mit je ca. 120 Seiten zum Seitenpreis von 38 Pfennig (Abbestellung jederzeit möglich).

Mehr Möglichkeiten mit Multiplan, WordStar, dBase, Lotus 1-2-3 und Erweiterungen

Dieses neue Nachschlagewerk bietet Ihnen u. a.:

- **Musteranwendungen zu Softwareengineering in dBase III**

Bausteine für ein komplettes Auftragsabwicklungssystem führen Ihnen die methodische Vorgehensweise bei der Einführung von DV-Lösungen vor Augen. Jeder Schritt von der Analyse bis zu fertigen Modulen für Ihre Stammdaten-, Lagerverwaltung und Fakturierung wird ausführlich erläutert;

- **raffinierte Lösungen mit dBase II und III;** an praktischen Beispielen erfahren Sie die optimale Vorgehensweise beim Erstellen, Laden und Auswerten von Dateien. Ein Programmierkurs ermöglicht Ihnen das Erstellen individueller Add-on-Programme;
- **ein Utility zur Verknüpfung von dBase-Dateien mit Multiplan-Templates,** z. B. für Rechnungen, Angebote, Mahnungen…
- **Tips, Tricks und Utilities zu Multiplan,** u. a. trigonometrische Funktionen, Verknüpfen von Arbeitsblättern, Druckersteuerung…
- **Präsentationsgrafik mit dem Multiplan-Grafikeditor;**
- **Modulbibliotheken mit Worksheets,** u. a. für kaufmännischen Soll-Ist-Vergleich;
- **programmierte Textverarbeitung mit Wordstar,** z. B. Textschablonen, Mixdruck;
- **Musterlösungen zu Lotus 1-2-3,** u. a. Vereinsverwaltung;
- **Ergänzungsausgaben zum Grundwerk** mit Lösungsvorschlägen, u. a. zum Mehrbenutzerbetrieb mit dBase III plus, Compilereinsatz sowie neuen Anwendertips und Dienstprogrammen zu aktuellen Software-Weiterentwicklungen.

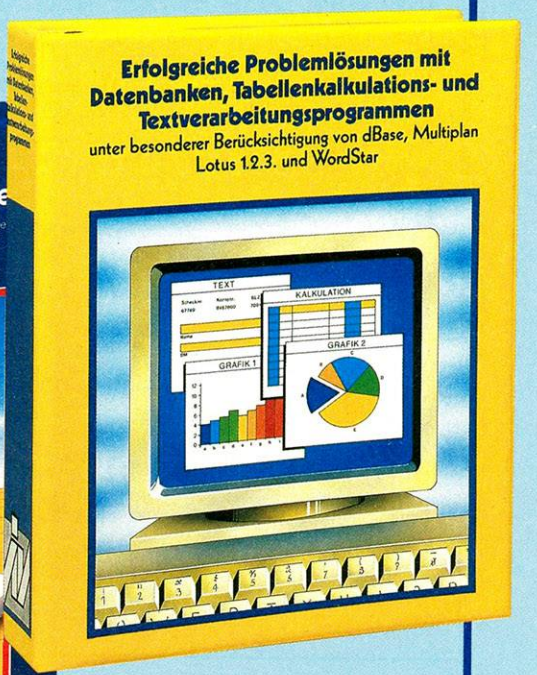
Fordern Sie noch heute an:

Erfolgreiche Problemlösungen mit Datenbanken, Tabellenkalkulations- und Textverarbeitungsprogrammen

unter besonderer Berücksichtigung von dBase, Multiplan, Lotus 1-2-3 und WordStar,

stabiler Ringbuchordner, Format DIN A4, Grundwerk ca. 550 Seiten, Bestell-Nr. 2700, zum Preis von DM 92,–

Alle 2–3 Monate erhalten Sie Ergänzungsausgaben zum Grundwerk mit je ca. 120 Seiten zum Seitenpreis von 38 Pfennig (Abbestellung jederzeit möglich).



```
[WTB]  736:   text('8',1,8,col[15]); txt(' y=SQR(x-z)',
           col[7]);
[SAE]  737:   text('9',1,9,col[15]); txt(' y=SQRT(x-z)'
           ,col[7]); WRITELN;
[PH7]  738:   WRITELN; WRITE(akey); READ(KBD,t);
[859]  739:   IF t IN ['1'..'9'] THEN BEGIN VAL(t,n,ck)
           ;
[OGL]  740:     GOTOXY(1,11); WRITE(spc); GOTOXY(19,n);
            WRITE('* SIN(q(xz))'); entveri; READ(KBD,t)
            ;
[PDA]  741:     IF t=#13 THEN sinus:=TRUE ELSE sinus:=F
           ALSE;
[F2A]  742:     FOR i:=1 TO pn DO BEGIN py[i]:=p[i][2];
[CT1]  743:       CASE n OF
[KX7]  744:         1,2,3 :q:=qxz(p[i][1],p[i][3]);
[GL7]  745:         4,5,6 :q:=(p[i][1]+p[i][3])/100;
[HM7]  746:         7,8,9 :q:=(p[i][1]-p[i][3])/100;
[TMO]  747:       END;
[CT1]  748:       CASE n OF
[U86]  749:         1,4,7 :p[i][2]:=ROUND(q*100);
[7W6]  750:         2,5,8 :p[i][2]:=ROUND(SQR(q)*20);
[898]  751:         3,6,9 :p[i][2]:=ROUND(SQRT(ABS(q))*
           200);
[TMO]  752:       END;
[TEH]  753:       IF sinus THEN p[i][2]:=ROUND(p[i][2]*
           SIN(qxz(p[i][1],p[i][3]))*0.5)
[TMO]  754:     END;
[4XA]  755:     IF sinus THEN GOTOXY(32,n) ELSE GOTOXY(
           19,n);
[H7B]  756:     WRITE('* y'); entveri; WRITE('
           '); READ(KBD,t);
[24H]  757:     IF t=#13 THEN FOR i:=1 TO pn DO p[i][2]
           :=ROUND(qxz(p[i][2],py[i])*100)
[TMO]  758:   END;
[TMO]  759: END;
[000]  760:
[YU8]  761: PROCEDURE print; VAR i,j :INTEGER;
[B21]  762: BEGIN
[736]  763:   text('Print ',1,1,col[7]);
[DOA]  764:   txt(nam,col[14]); txt('  1',col[15]);
[D19]  765:   txt('(data)',col[7]); txt(' 2',col[15]);
[UBC]  766:   txt('(points only)'+okey,col[7]); READ(KB
           D,t);
[2S3]  767:   IF t='1' THEN BEGIN
[3C8]  768:     WRITELN(LST,nam); WRITELN(LST);
[MN5]  769:     FOR i:=1 TO en DO BEGIN
[RB6]  770:       WRITE(LST,i,'   (',e[i],') ');
[5PA]  771:       FOR j:=1 TO 3 DO WRITE(LST,p[ABS(e[i]
           )][j],' ');
[VR2]  772:       WRITELN(LST)
[TMO]  773:     END;
[TMO]  774:   END;
[SN2]  775:   IF t='2' THEN
[7XG]  776:     FOR i:=1 TO pn DO WRITELN(LST,i,' (',p[
           i][1],' ',p[i][2],' ',p[i][3],')')
[TMO]  777: END;
[000]  778:
[155]  779: PROCEDURE fragende;
[B21]  780: BEGIN
[G4A]  781:   txt('Press ',col[7]); txt('ESC ',col[15])
           ;
[MN8]  782:   txt('to quit program',col[7]);
[N5D]  783:   text('any other key to continue',1,2,col[
           7]);
[YM8]  784:   READ(KBD,t); IF t=#27 THEN ende:=TRUE
[TMO]  785: END;
[000]  786:
[DC7]  787: PROCEDURE vzchn(i :INTEGER);
[XR9]  788: BEGIN IF (i>0) THEN txt('+',col[14]) END;
[000]  789:
[RC5]  790: PROCEDURE textinit;
[3FD]  791: BEGIN TEXTCOLOR(col[14]); cursor(FALSE); CL
           RSCR END;
[000]  792:
[TU4]  793: PROCEDURE namaus;
[P5E]  794: BEGIN text('         ',3,4,col[14]); text(na
           m,3,4,col[14]) END;
[000]  795:
[BU4]  796: PROCEDURE zeile6;
[B21]  797: BEGIN
[2P5]  798:   text(spc,3,6,col[0]);
[05L]  799:   text('points(',3,6,col[7]); wrt(pn); txt(
           ') edges(',col[7]); wrt(en);
[1XC]  800:   txt(') shapes(',col[7]); wrt(sn); txt(')
           ',col[7])
[TMO]  801: END;
[000]  802:
[TX4]  803: PROCEDURE zeile23;
[B21]  804: BEGIN
[FR5]  805:   text(spc,3,23,col[0]);
[WXC]  806:   text('0',3,23,col[15]); txt('ptics(',col[
           7]); wrt(b);
[L4F]  807:   txt(')  de',col[7]); txt('G',col[15]); tx
           t('rees x(',col[7]);
[ODA]  808:   wrt(w[1]); txt('°)y(',col[7]); wrt(w[2]);
[OOE]  809:   txt('°)z(',col[7]); wrt(w[3]); txt('°)  '
           ,col[7]);
[TFE]  810:   txt('Z',col[15]); txt('oom',col[7]); txt(
           '±',col[15]);
[B89]  811:   txt('(',col[7]); vzchn(zo); wrt(zo);
[U4F]  812:   txt(')  or',col[7]); txt('I',col[15]); tx
           t('gin x(',col[7]);
[U8E]  813:   wrt(ox); txt(')y(',col[7]); wrt(oy); txt(
           ')',col[7]);
[TMO]  814: END;
[000]  815:
[GT3]  816: PROCEDURE cls;
[2Y5]  817: BEGIN CLRSCR; winoff END;
[000]  818:
[LVD]  819: PROCEDURE boshow(spalte,zeile :INTEGER; bs:
           CHAR);
[B21]  820: BEGIN
[6CE]  821:   TEXTCOLOR(col[14]); GOTOXY(spalte,zeile);
            WRITE(bs);
[PB7]  822:   TEXTCOLOR(col[7]); GOTOXY(3,6)
[TMO]  823: END;
[000]  824:
[RK8]  825: PROCEDURE menue; VAR n :INTEGER;
[B21]  826: BEGIN
[H13]  827:   textinit;
[M6A]  828:   txt(lin2o,col[1]); txt('‖',col[1]);
[WUA]  829:   txt(' SCad 6.2    ',col[7]); txt('| ',col
           [1]);
[U89]  830:   txt('B',col[15]); txt('uild   ',col[7]);
[30D]  831:   txt('S',col[15]); txt('ave   ',col[7]); t
           xt('E',col[15]);
[6ZC]  832:   txt('d',col[7]); txt('!',col[15]); txt('t
           ',col[7]);
[RA9]  833:   txt('P',col[15]); txt('lace   ',col[7]);
[DLC]  834:   txt('T',col[15]); txt('urn  ',col[7]); tx
           t('R',col[15]);
[TMB]  835:   txt('otation  s',col[7]); txt('H',col[15]
           );
[17H]  836:   txt('ow(',col[7]); txt(bochk(shw),col[14]
           ); txt(')  ',col[7]);
[FGD]  837:   txt('Q',col[15]); txt('uit   ',col[7]); t
           xt('‖',col[1]);
[9N6]  838:   txt('‖              | ',col[1]);
[A7E]  839:   txt('V',col[15]); txt('ie',col[7]); txt('
           W',col[15]);
[N8K]  840:   txt('s   ',col[7]); txt('L',col[15]);txt(
           'oad   points  ',col[7]);
[S8D]  841:   txt('F',col[15]); txt('orm    ',col[7]);
           txt('A',col[15]);
[2RF]  842:   txt('dd   ',col[7]); txt('D',col[15]); tx
           t('eform    ',col[7]);
[HNF]  843:   txt('C',col[15]); txt('hange   pr',col[7]
           ); txt('J',col[15]);
[9HA]  844:   txt('pol ',col[7]); txt('‖',col[1]);
[9N6]  845:   txt('‖              | ',col[1]);
[8A9]  846:   txt('ke',col[7]); txt('Y',col[15]);
[6SA]  847:   txt('(',col[7]); txt(bochk(key),col[14]);
[429]  848:   txt(')  ',col[7]); txt('M',col[15]);
[KME]  849:   txt('erge  edges   shapes  ',col[7]); txt
           ('K',col[15]);
[82G]  850:   txt('ill  f',col[7]); txt('U',col[15]); t
           xt('nction  a',col[7]);
[CX8]  851:   txt('X',col[15]); txt('es(',col[7]);
[P9B]  852:   txt(bochk(axs),col[14]); txt(')  o',col[7
           ]);
[K8D]  853:   txt('N',col[15]); txt('edge ',col[7]); tx
           t('‖',col[1]);
[ULG]  854:   txt(lin1o,col[1]); text('‖',1,6,col[1]);
           text('‖',80,6,col[1]);
[U65]  855:   txt(lin1u,col[1]);
[AAK]  856:   FOR n:=8 TO 21 DO BEGIN text('‖',1,n,col[
           1]); text('‖',80,n,col[1]); END;
[RKG]  857:   txt(lin1u,col[1]); txt('‖ ',col[1]); text
           ('‖',80,23,col[1]);
[U65]  858:   txt(lin2u,col[1]);
[N27]  859:   namaus; zeile6; zeile23;
[TMO]  860: END;
[000]  861:
[GK5]  862: PROCEDURE animation;
[MH1]  863: LABEL 0;
[M64]  864: CONST amax         =60;
[VN2]  865:       wax          =180;
[N2C]  866: VAR   ua,va        :ARRAY[1..amax,1..wax] OF
            INTEGER;
[KM3]  867:       wa           :punkt;
[064]  868:       i,j,n        :INTEGER;
[B21]  869: BEGIN
[RY1]  870:   winon;
[409]  871:   IF (pn>amax) OR (en>amax) THEN BEGIN
[P5D]  872:     txt(memo,col[4]); REPEAT UNTIL KEYPRESS
           ED; cls;
[3B3]  873:   END ELSE BEGIN
```

```
[GZ6]  874:      FOR n:=1 TO 3 DO wa[n]:=w[n];
[ULF]  875:      GOTOXY(1,1); WRITE('Phase'); txt('
         ...please wait',col[4]);
[54B]  876:      FOR j:=1 TO wax DO BEGIN GOTOXY(7,1); w
         rt(j);
[V69]  877:        FOR n:=1 TO 3 DO wa[n]:=wa[n]+2 MOD 3
         60;
[4W6]  878:        matr3d(wa[1],wa[2],wa[3]);
[MN5]  879:        FOR i:=1 TO en DO BEGIN
[FEF]  880:          pro2d(p[ABS(e[i])][1],p[ABS(e[i])][
         2],p[ABS(e[i])][3]);
[SZ6]  881:          ua[i,j]:=u1; va[i,j]:=v1;
[TMO]  882:        END;
[TMO]  883:      END;
[A44]  884:      egamode(TRUE);
[XG1]  885:      REPEAT
[N66]  886:        FOR j:=1 TO wax DO BEGIN
[DU1]  887:          clega;
[MN5]  888:          FOR i:=1 TO en DO BEGIN
[HT6]  889:            u1:=ua[i,j]; v1:=va[i,j];
[1MA]  890:            IF (e[i]>0) THEN line(u2,v2,u1,v1
         ,col[9]);
[MS3]  891:            u2:=u1; v2:=v1;
[AXD]  892:            IF KEYPRESSED THEN BEGIN READ(KBD
         ,t); IF (t=#27) THEN GOTO 0; END;
[TMO]  893:          END;
[TMO]  894:        END;
[9M2]  895:      UNTIL (t=#27);
[839]  896: 0:   egamode(FALSE); winoff; menue
[TMO]  897:    END;
[TMO]  898: END;
[000]  899:
[YX4]  900: PROCEDURE inout;
[B21]  901: BEGIN
[1T9]  902:    GOTOXY(3,8); TEXTCOLOR(col[7]); READ(KBD,
         t);
[GU1]  903:    CASE t OF
[06D]  904:           '+' :IF zo<10 THEN BEGIN zo:=SUCC(z
         o); size; zeile23 END;
[34D]  905:           '-' :IF zo>-10 THEN BEGIN zo:=PRED(
         zo); size; zeile23 END;
[ONG]  906:      '0','o' :BEGIN b:=SUCC(b) MOD 11; IF b=
         0 THEN b:=1; size; zeile23 END;
[ZE9]  907:      'Z','z' :BEGIN zo:=0; size; zeile23 END
         ;
[LND]  908:      'Y','y' :BEGIN key:=NOT key; boshow(21,
         4,bochk(key)) END;
[7SD]  909:      'X','x' :BEGIN axs:=NOT axs; boshow(69,
         4,bochk(axs)) END;
[8TF]  910:      'B','b' :BEGIN egamode(TRUE); views; eg
         amode(FALSE); menue END;
[A13]  911:      'L','l' :BEGIN
[B89]  912:                 winon; dir; WRITE('Load'); n
         amein;
[D9D]  913:                 load; IOer:=FALSE; cls; nama
         us; zeile6; zeile23
[TMO]  914:               END;
[233]  915:      'S','s' :BEGIN
[R1F]  916:                 winon; dir; WRITE('Save'); n
         amein; save; cls; namaus
[TMO]  917:               END;
[KZ2]  918:      'E','e' :BEGIN
[D3F]  919:                 winon; points; cls; zeile6;
         winon; edges; cls; zeile6
[TMO]  920:               END;
[FXD]  921:      'Q','q' :BEGIN winon; fragende; IF NOT
         ende THEN cls END;
[BVB]  922:      'G','g' :BEGIN winon; degrees; cls; zei
         le23 END;
[AS9]  923:      'I','i' :BEGIN ox:=320; oy:=100; zeile2
         3 END;
[UDC]  924:      'J','j' :BEGIN winon; polyeder; cls; ze
         ile6 END;
[9X9]  925:      '?'     :BEGIN winon; changedir; cls EN
         D;
[TMO]  926:    END;
[KE4]  927:    IF (sn>0) THEN BEGIN
[GU1]  928:      CASE t OF
[U33]  929:        'V','v' :BEGIN
[BED]  930:                   egamode(TRUE); matr3d(w[1]
         ,w[2],w[3]); draw3d(FALSE);
[246]  931:                   egamode(FALSE); menue
[TMO]  932:                 END;
[243]  933:        'W','w' :BEGIN
[77D]  934:                   egamode(TRUE); matr3d(w[1]
         ,w[2],w[3]); draw3d(TRUE);
[ZY6]  935:                   egamode(FALSE); menue END;
[K13]  936:        'M','m' :BEGIN
[6MB]  937:                   winon; dir; WRITE('Merge')
         ; namein; merge;
[3N9]  938:                   IOer:=FALSE; cls; namaus;
         zeile6
[TMO]  939:                 END;
[ZBD]  940:        'P','p' :BEGIN winon; einplace; place
         form; cls END;
[HSC]  941:        'F','f' :BEGIN winon; einform; placef
         orm; cls END;
[32B]  942:        'C','c' :BEGIN winon; exchange(1,pn);
          cls END;
[LMA]  943:        'D','d' :BEGIN winon; fractal(1,pn);
         cls END;
[MSB]  944:        'A','a' :BEGIN winon; addiere; cls; z
         eile6 END;
[55B]  945:        'K','k' :BEGIN winon; kill; cls; zeil
         e6 END;
[VHC]  946:        'R','r' :BEGIN winon; rotation; cls;
         zeile6 END;
[3SA]  947:        'U','u' :BEGIN winon; funktionen; cls
          END;
[VXA]  948:        'T','t' :BEGIN winon; turnaround; cls
          END;
[DNB]  949:        'N','n' :BEGIN winon; onedge; cls; ze
         ile6 END;
[NLD]  950:        'H','h' :BEGIN shw:=NOT shw; boshow(6
         9,2,bochk(shw)) END;
[U2C]  951:    {PrtSc} '*' :BEGIN winon; print; cls END;
[SYA]  952:            '!' :BEGIN winon; delins; cls; ze
         ile6 END;
[FL4]  953:            '/' :animation;
[TMO]  954:      END;
[TMO]  955:    END;
[TMO]  956: END;
[000]  957:
[S99]  958: PROCEDURE vorspann; VAR n :INTEGER;
[B21]  959: BEGIN
[FF3]  960:    n:=PARAMCOUNT;
[1C3]  961:    IF n=0 THEN BEGIN
[9WB]  962:      text('╔═══════════════════════════════
         ═══════════════════════',13,10,col[1]);
[MVC]  963:      text('||                              
                                ||',13,11,col[1]);
[ESD]  964:      text('||                                
                                  ',13,12,col[1]); txt
         (vers,col[15]);
[BE5]  965:      txt('                                 
                                ||',col[1]);
[YVC]  966:      text('||                              
                                ||',13,13,col[1]);
[28A]  967:      text('||------------------------------
         ----------------------||',13,14,col[1]);
[WR7]  968:      text('||                              
                                  ',13,15,col[1]);
[G7A]  969:      txt('Press any key to start',col[7]);
[366]  970:      txt('                                 
                                ||',col[1]);
[6XB]  971:      text('╚═══════════════════════════════
         ═══════════════════════',13,16,col[1]);
[BL5]  972:      REPEAT UNTIL KEYPRESSED;
[3R6]  973:      nam:='?'; pn:=0; en:=0; sn:=0;
[3B3]  974:    END ELSE BEGIN
[LC8]  975:      nam:=PARAMSTR(1); namup; load;
[2NF]  976:      IF IOer THEN BEGIN pn:=0; en:=0; sn:=0;
          nam:='?'; IOer:=FALSE END;
[TMO]  977:    END;
[TMO]  978: END;
[000]  979:
[B21]  980: BEGIN
[M7A]  981:    TEXTMODE(3); TEXTBACKGROUND(0); textinit;
[GLA]  982:    akt:=11; nam1:=''; w[1]:=0; w[2]:=0; w[3]
         :=0;
[6E7]  983:    ox:=320; oy:=175; b:=1; zo:=0; size;
[T5F]  984:    ende:=FALSE; IOer:=FALSE; key:=FALSE; axs
         :=FALSE; shw:=FALSE;
[WS9]  985:    vorspann; pn1:=pn; en1:=en; sn1:=sn;
[N68]  986:    loeschfeld; resetfh; menue;
[XG1]  987:    REPEAT
[CO2]  988:      inout;
[FF2]  989:    UNTIL ende;
[SLB]  990:    IF (sn<>sn1) OR (en<>en1) OR (pn<>pn1) TH
         EN BEGIN
[HKC]  991:      text('Save changes',1,4,col[7]); namein
         ; save
[TMO]  992:    END;
[226]  993:    winoff; cursor(TRUE);
[SK1]  994:    CLRSCR
[5LO]  995: END.
```

Listing 2. SCad trägt professionelle Züge

Scroll-Routine für beliebige Bildschirmfenster

Herkömmliche Scrollroutinen beziehen sich meist auf den gesamten Bildschirminhalt. Möchten Sie aber bestimmte Bildausschnitte bewegen, stehen Sie stets vor einem größeren Programmieraufwand. Die hier vorgestellte Routine nimmt Ihnen diese Arbeit ab.

So manchem Programmierer bereitete es Schwierigkeiten, mehrere Textzeilen (mehr als 25) auf den Bildschirm zu bringen. Gilt es nun, einen Editor zu programmieren oder lange Zahlenkolonnen am Bildschirm darzustellen, so stehen dem Programmierer zwei Lösungen zur Verfügung.

Die eine Methode beruht darauf, den Bildschirm zu füllen, anschließend zu löschen und wieder von vorne zu beginnen. Die zweite Methode liegt in der automatischen Scroll-Funktion des Bildschirms. Wird über die letzte Zeile hinausgeschrieben (zum Beispiel mit »writeln«), so scrollt der Bildschirminhalt automatisch eine Zeile nach oben.

Alle diese Methoden haben aber einen Nachteil: Sie beeinflussen immer den gesamten Bildschirm. Legen Sie aber Wert darauf, innerhalb eines umrandeten Bildbereichs Texte zu scrollen, stoßen Sie auf schwerwiegende Probleme.

Aus diesem Grund entstanden die beiden Turbo Pascal-Prozeduren »ScrollDown« und »ScrollUp« (Listing 1). Mit Hilfe dieser Routinen sind Sie in der Lage, beliebige Bildschirmausschnitte sowohl nach oben als auch nach unten scrollen zu lassen.

Die Routinen werden mit folgenden Parameterangaben aufgerufen:

```
ScrollDown (XLu,YLu,XRd,YRd,Line,Attribut);
```

beziehungsweise

```
ScrollUp (XLu,YLu,XRd,YRd,Line,Attribut);
```

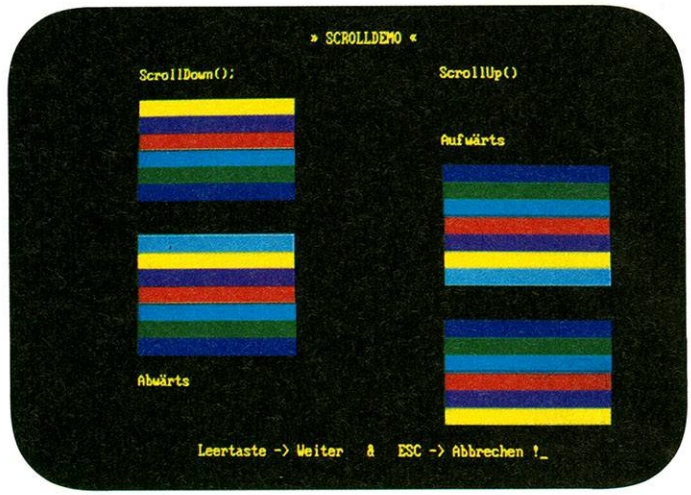


Bild. Die Farbbalken der beiden Bildschirmfenster werden mit den Scroll-Routinen bewegt

In den Parametern »XLu« und »YLu« stehen die Koordinaten der linken oberen Ecke des Bildschirmfensters. »XLu« darf Werte zwischen 1 und 80 annehmen, »YLu« hingegen maximal den Wert 25. Analog hierzu übergeben die Parameter »XRd« und »YRd« die Position der rechten unteren Fensterecke. Auch hier darf der Parameter »XRd« nicht größer als 80 sein. Sein Minimum legt der Parameter »XLu« fest.

»YRd« dürfen Sie Werte zwischen »YLu« und 25 zuordnen.

Die Zahl der zu verschiebenden Zeilen wird mit dem Parameter »Line« festgelegt. Normalerweise steht hier der Wert 1. Der Parameter »Attribut« übergibt die aktuelle Farbe der zu verschiebenden Zeile.

Im Beispielprogramm »demo.pas« (Listing 2) dienen die beiden Prozeduren »ScrollDown« und »ScrollUp« gleichzeitig dazu, in zwei getrennten Fenstern bunte Farbbalken sowohl nach oben als auch nach unten zu scrollen (Bild).

Ein Druck auf die Leertaste löst den Scroll-Vorgang aus. Die Esc-Taste beendet das Programm. Sie finden sicher viele Anwendungsbereiche, für die sich diese beiden Prozeduren bestens bewähren.

(Thomas Kowalski/sp)

```
[ASO]    0: TYPE
[U54]    1:   T_REGISTER=RECORD
[SM3]    2:                  CASE INTEGER OF
[6CA]    3:                    1: (AX,BX,CX,DX,BP,SI,DI,D
           S,ES,FLAGS: INTEGER);
[BZ6]    4:                    2: (AL,AH,BL,BH,CL,CH,DL,D
           H: BYTE);
[TMO]    5:                END;
[000]    6:
[EEO]    7: VAR
[KDB]    8:   Reg:        T_REGISTER;          (* 8088/
           8086 Register *)
[000]    9:
[000]   10:
[R8G]   11: (* Bewirkt das Aufwärtsscrollen des Bildsch
           irmes *)
[000]   12:
[K1E]   13: PROCEDURE ScrollUp(XLu,YLu,XRd,YRd,Line,Att
           r: BYTE);
[B21]   14: BEGIN
[ED3]   15:   Attr:=Attr*16;
[CE2]   16:   XLu:=XLu-1;
[WPE]   17:   YLu:=YLu-1;                          (* Aufbe
           reitung der Parameter *)
```

```
[FE2]   18:   XRd:=XRd-1;
[PE2]   19:   YRd:=YRd-1;
[H15]   20:   inline($50/$53/$51/$52/
[C6D]   21:          $56/$57/$1E/$06/$FB);      (* Regis
           ter werden gesichert *)
[6D2]   22:   Reg.AH:=$06;
[H2L]   23:   Reg.AL:=Line;                        (* Anzah
           l        der zu verschiebenden Zeilen *)
[OON]   24:   Reg.BH:=Attr;                        (* Farba
           ttribute der zu verschiebenden Zeilen *)
[TBF]   25:   Reg.CH:=YLu;                         (* Y-Pos
           ition    linke oberer Ecke *)
[BCF]   26:   Reg.CL:=XLu;                         (* X-Pos
           ition    linke oberer Ecke *)
[1AG]   27:   Reg.DH:=YRd;                         (* Y-Pos
           ition    rechte untere Ecke *)
[LAG]   28:   Reg.DL:=XRd;                         (* X-Pos
           ition    rechte untere Ecke *)
[35H]   29:   Intr($10,Reg);                       (* BIOS
           Interruptnr. $10 wird aufgerufen *)
[GN6]   30:   inline($07/$1F/$5F/$5E/$5A/$59/
[T3H]   31:          $5B/$58/$8B/$E5/$D5/$CF); (* Regis
           ter werden wieder hergestellt *)
[TMO]   32: END;
```

```
[000]   33:
[000]   34:
[ASF]   35: (* Bewirkt das Abwärtsscrollen des Bildschi
          rmes *)
[000]   36:
[Y6F]   37: PROCEDURE ScrollDown(XLu,YLu,XRd,YRd,Line,A
          ttr: BYTE);
[B21]   38: BEGIN
[ED3]   39:   Attr:=Attr*16;
[CE2]   40:   XLu:=XLu-1;
[WPE]   41:   YLu:=YLu-1;                          (* Aufbe
          reitung der Parameter *)
[FE2]   42:   XRd:=XRd-1;
[PE2]   43:   YRd:=YRd-1;
[H15]   44:   inline($50/$53/$51/$52/
[C6D]   45:          $56/$57/$1E/$06/$FB);         (* Regis
          ter werden gesichert *)
[9D2]   46:   Reg.AH:=$07;
[H2L]   47:   Reg.AL:=Line;                        (* Anzah
          l          der zu verschiebenden Zeilen *)
[00N]   48:   Reg.BH:=Attr;                        (* Farba
          ttribute der zu verschiebenden Zeilen *)
[TBF]   49:   Reg.CH:=YLu;                         (* Y-Pos
          ition    linke oberer Ecke *)
[BCF]   50:   Reg.CL:=XLu;                         (* X-Pos
          ition    linke oberer Ecke *)
[1AG]   51:   Reg.DH:=YRd;                         (* Y-Pos
          ition    rechte untere Ecke *)
[LAG]   52:   Reg.DL:=XRd;                         (* X-Pos
          ition    rechte untere Ecke *)
[35H]   53:   Intr($10,Reg);                       (* BIOS
          Interruptnr. $10 wird aufgerufen *)
[GN6]   54:   inline($07/$1F/$5F/$5E/$5A/$59/
[T3H]   55:          $5B/$58/$8B/$E5/$D5/$CF); (* Regis
          ter werden wieder hergestellt *)
[TMO]   56: END;
```

Listing 1. Quellcode der beiden Prozeduren »ScrollDown« und »ScrollUp«

```
[94L]    0: PROGRAM ScrollDemo(Input, Output); (* Demop
           rogramm für das Auf- *)
[UZ8]    1:                                    (* bzw.
           Abwärtssrollen des   *)
[A47]    2:                                    (* Bilds
           chirmes.             *)
[000]    3:
[XA5]    4: {$I scroll.pas}
[000]    5:
[EE0]    6: VAR
[YA4]    7:    Color:      Byte;
[9C3]    8:    Key:        Char;
[KX7]    9: (* Hauptanweisungsteil *)
[000]   10:
[ZA4]   11: PROCEDURE Main;
[B21]   12: BEGIN
[KW4]   13:   ClrScr; Color:=0;
[9K9]   14:   GotoXY(33,1); write('» SCROLLDEMO «');
[BS9]   15:   GotoXY(10,3); write('ScrollDown();');
[RA8]   16:   GotoXY(50,3); write('ScrollUp()');
[LO8]   17:   GotoXY(10,5); write('Abwärts');
[L58]   18:   GotoXY(50,23); write('Aufwärts');
[PTG]   19:   GotoXY(18,25); write('Leertaste -> Weiter
              &   ESC -> Abbrechen !');
[XG1]   20:   REPEAT
[GP7]   21:     ScrollDown(10,5,30,23,1,Color);
[WM6]   22:     ScrollUp(50,5,70,23,1,Color);
[BH3]   23:     Read(KBD,Key);
[Z9C]   24:     IF Color<=7 THEN Color:=Color+1 ELSE Co
          lor:=0;
[XN3]   25:   UNTIL Key=CHR(27);
[NF2]   26:   ClrScr;
[TMO]   27: END;
[000]   28:
[B21]   29: BEGIN
[991]   30:   Main;
[5LO]   31: END.
```

Listing 2. Demonstrationsprogramm der Scroll-Funktion

W E N D I N – D O S

Das Multitasking, Multiuser Betriebssystem

- *für IBM-PC, XT, AT, 80386 und Kompatible*
- *MS-DOS Programme unter WENDIN-DOS lauffähig*
- *und viele Features mehr, die von einem Multitasking/-user Betriebssystem erwartet werden*

WENDIN-DOS.....DM 235,-
WENDIN-DOS Application Dev. Kit (mit Source).DM 235,-

weitere WENDIN-Produkte incl. Source Code
PCVMS (VAX/VMS Environment).....DM 235,-
PCNX (UNIX Environment).....DM 235,-
OPERATING SYSTEM TOOLBOX.....DM 235,-
XTC (Editor für Programmierer).....DM 199,-

P C G L O B E

Grafik und Fakten über 177 Länder und 5 Kontinente auf Ihrem Monitor

für IBM-PC, XT, AT und Kompatible mit CGA für DM 150,-

P O P U P R e f e r e n c e G u i d e s

Residente 'Manuals' für schnelle und effektive Programmentwicklung

- *für IBM-PC, XT, AT und Kompatible sind folgende Pop Up Reference Guides verfügbar:*

Turbo Pascal 3.0.....DM 170,-
Turbo Prolog 1.1.....DM 170,-
Diverse 'C'-Compiler.....DM 170,-
Programmer's Reference.....DM 170,-
MS-DOS.....DM 99,-
User Data-Base Compiler.....DM 140,-
(für die Erstellung eigener Guides)

Utilities
ISAM/BTree....DM 289
"+M.-user Sup.DM 425
PC Tools Del..DM 199
Tree Diagram..DM 180
Source Print..DM 210

CAD
Drafix....ab DM 2000,-
ModelMate.....DM 1199,-

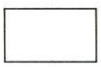
Viviane Wolff, Dipl.-Ing.
EDV-Vertrieb
Leo-Graetz-Str. 7
8000 München 70
Telex: Btx 0897809608
Tel.: 089/780 96 08

Demos (DM 5,-) auf Anfrage
Es gelten die jew. Monatspreise incl. Mwst, zzgl. Versand.

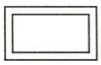
Schnelle Rahmen-Routine für jeden Zweck

Viele Anwender-Programme machen durch Bildschirmfenster mit Rahmen einen erheblich professionelleren Eindruck. Der langwierigen Zeichnerei von Rahmen, das mit Hochsprachen wie Turbo Pascal sehr langsam vor sich geht, macht dieses Assembler-Programm ein wenig Beine.

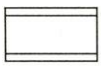
Mit der Assembler-Routine Frame1 bringen Sie einen Rahmen zur Verschönerung Ihrer Turbo Pascal-Programme sehr schnell auf den Bildschirm. Dabei kommen BIOS-Routinen zur Anwendung, so daß dieses Programm auf allen PCs (XT oder AT) läuft, egal welche Grafikkarte eingebaut ist. Die Geschwindigkeit liegt beinah so hoch, wie das beim direkten Schreiben in den Bildschirmspeicher geschieht. Die Assembler-Routine kann sowohl mit Hochsprachen wie Turbo Pascal als auch mit Assembler-Programmen genutzt werden.
Frame1 benötigt nur sechs Parameter, um einwandfrei zu arbeiten:
yx1 – linke obere x-Koordinate des Rahmens (0 bis 78)
x2 – rechte untere x-Koordinate des Rahmens (0 bis 79)
y1 – linke obere y-Koordinate des Rahmens (0 bis 24)
y2 – rechte untere y-Koordinate des Rahmens (0 bis 25)
attribut – Bildschirmattribut (0 = NE, 2 = NF, 10 = BF, 11 = RV)
typ – Rahmentyp (1 bis 4)
Folgenden Rahmentypen stehen zur Auswahl:



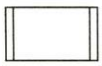
Typ 1: Rahmen mit einfacher Linie.



Typ 2: Rahmen mit doppelter Linie.



Typ 3: doppelte Linie horizontal, einfache Linie vertikal



Typ 4: einfache Linie horizontal, doppelte Linie vertikal

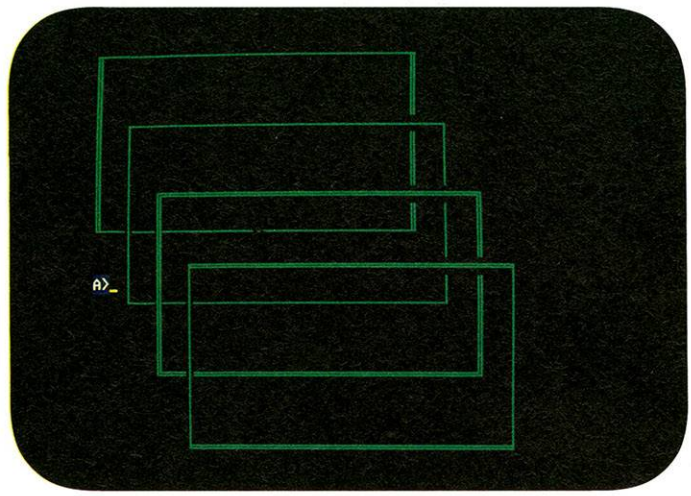


Bild. Vier Rahmentypen überlagern sich

Benötigen Sie die Prozedur in einem Turbo Pascal-Programm, so muß diese als »external« deklariert werden. Das sieht folgendermaßen aus:

```
procedure frame1(x1,x2,y1,y2,attr,typ:byte); external 'frame1.bin';
```

Soll die Rahmen-Routine in einem Assembler-Programm zum Einsatz kommen, sind folgende Vorbereitungen nötig:

```
push x1          push attribut
push x2          push typ
push y1          call frame1
push y2
```

Die Datei »frame1.asm« wurde mit dem Microsoft Makro-Assembler (MASM) erstellt. Listing 1 läßt sich mit dem MASM eintippen, assemblieren und linken. Das Programm »Exe2Bin« erzeugt anschließend die Datei »frame1.bin«. Besitzen Sie diesen Assembler nicht, so tippen Sie die Datei »frame1.bin« einfach als Hexdump ab (Listing 2). Die Datei »frame1.pas« (Listing 3) ist ein Turbo Pascal-Programm, das Ihnen die Fähigkeiten der Rahmen-Routine demonstriert (Bild).

(Jörg Dirbach/sp)

```
[ZO7]    0: ;*****************************************
         *******
[FDA]    1: ;* Assembler-Routine  FRAME1.ASM
              *
[XE7]    2: ;* ----------------------------------------
         ----- *
[4N7]    3: ;* Procedure
              *
[BVC]    4: ;* frame1(x1,x2,y1,y2,attribut,typ:byte);
              *
[6TB]    5: ;* x1        : x-Koordinate links oben  (0..
         78)  *
[WTC]    6: ;* x2        : x-Koordinate rechts unten (1.
         .79)  *
[6RB]    7: ;* y1        : y-Koordinate links oben (0..2
         4)    *
[FSC]    8: ;* y2        : x-Koordinate rechts unten (1.
         .25)  *
[74C]    9: ;* attribut : Bildschirmattribut
              *
[ET7]   10: ;*              (z.B NE:0,NF:2,BF:10,RV:112)
              *
[9L9]   11: ;* typ       : Rahmentyp (1..4)
              *
[B8E]   12: ;* für den Aufruf von Turbo Pascal oder ein
         em    *
[MBB]   13: ;* anderen Assembler-Programm
              *
[XE7]   14: ;* ----------------------------------------
         ----- *
[5F8]   15: ;* von Turbo Pascal
              *
[A1B]   16: ;*  Vorbereiten der Routine :
              *
[CF7]   17: ;*       Masm frame1;
              *
[4F7]   18: ;*       Link frame1;
              *
```

```
[AA8]   19: ;*       Exe2Bin frame1;
              *
[G1F]   20: ;*  Deklaration im Turbo Pascal-Programm mi
           t:
[NS7]   21: ;*  Procedure
              *
[NOC]   22: ;*       FRAME1(x1,x2,y1,y2,attribut,typ:b
         yte); *
[358]   23: ;*       EXTERNAL 'FRAME1.BIN';
              *
[AX7]   24: ;*  Aufruf mit :
              *
[11B]   25: ;*  FRAME1(x1,x2,y1,y2,attribut,typ);
              *
[AF7]   26: ;* ----------------------------------------
         ------*
[5L8]   27: ;* von Assembler:
              *
[N7C]   28: ;*  Parameterübergabe und Aufruf:
              *
[AV6]   29: ;*   push x1
              *
[EV6]   30: ;*   push x2
              *
[DV6]   31: ;*   push y1
              *
[HV6]   32: ;*   push y2
              *
[XN8]   33: ;*   push attribut
              *
[EG7]   34: ;*   push typ
              *
[S97]   35: ;*   call FRAME1
              *
[LMD]   36: ;*   Beim Aufruf werden alle verwendeten
              *
[6VD]   37: ;*     Register gerettet und bei der Rückke
         hr    *
```

```
[9YC]   38: ;*      ins aufrufende Programm wird der Sta
          ck     *
[CD7]   39: ;*      gereinigt
                 *
[XE7]   40: ;* ---------------------------------------
          -----  *
[33B]   41: ;* written by Joerg Dirbach 13.09.1987
                 *
[Z07]   42: ;*****************************************
          *******
[000]   43:
[R84]   44: title   frame1.asm
[C22]   45: page    66,96
[000]   46:
[D43]   47: code    SEGMENT
[FG3]   48: PUBLIC  frame1
[WF4]   49:         ASSUME  cs:code
[000]   50:
[M54]   51: frame1  PROC  near
[000]   52:
[6P8]   53:         ;benutzte Register retten
[K76]   54:         push  ax            ;rette ax
[W76]   55:         push  bx            ;rette bx
[686]   56:         push  cx            ;rette cx
[G86]   57:         push  dx            ;rette dx
[V3B]   58:         push  bp            ;rette basis poi
          nter
[000]   59:
[WTA]   60:         ;basepointer mit stack definieren
[E55]   61:         mov   bp,sp         ;bp=sp
[000]   62:
[W1C]   63:         ;Zeichensatz auf Stack initialisier
          en
[AWB]   64:         mov   ax,[bp+12]    ;zeichensatz in
          ax laden
[PB8]   65:         cmp   ax,1          ;zeichensatz 1 ?
[ZV8]   66:         jnz   zeich2        ;zeichensatz<>1
[5B4]   67:         mov   ax,218        ;┌
[6C3]   68:         push  ax            ;
[S64]   69:         mov   ax,191        ;┐
[6C3]   70:         push  ax            ;
[174]   71:         mov   ax,192        ;└
[6C3]   72:         push  ax            ;
[WA4]   73:         mov   ax,217        ;┘
[6C3]   74:         push  ax            ;
[P54]   75:         mov   ax,179        ;│
[6C3]   76:         push  ax            ;
[584]   77:         mov   ax,196        ;─
[6C3]   78:         push  ax            ;
[ZAB]   79:         jmp   z_end         ;zeichenzatz 1 g
          eladen
[000]   80:
[EKA]   81: zeich2: cmp   ax,2          ;zeichensatz 2 ?
[5W8]   82:         jnz   zeich3        ;zeichensatz<>2
[H74]   83:         mov   ax,201        ;╔
[6C3]   84:         push  ax            ;
[T64]   85:         mov   ax,187        ;╗
[6C3]   86:         push  ax            ;
[874]   87:         mov   ax,200        ;╚
[6C3]   88:         push  ax            ;
[274]   89:         mov   ax,188        ;╝
[6C3]   90:         push  ax            ;
[G64]   91:         mov   ax,186        ;║
[6C3]   92:         push  ax            ;
[N84]   93:         mov   ax,205        ;═
[6C3]   94:         push  ax            ;
[4AB]   95:         jmp   z_end         ;zeichensatz 2 g
          eladen
[000]   96:
[RKA]   97: zeich3: cmp   ax,3          ;zeichensatz 3 ?
[BW8]   98:         jnz   zeich4        ;zeichensatz<>3
[S94]   99:         mov   ax,213        ;╒
[6C3]  100:         push  ax            ;
[Y54]  101:         mov   ax,184        ;╕
[6C3]  102:         push  ax            ;
[F94]  103:         mov   ax,212        ;╘
[6C3]  104:         push  ax            ;
[F64]  105:         mov   ax,190        ;╛
[6C3]  106:         push  ax            ;
[P54]  107:         mov   ax,179        ;│
[6C3]  108:         push  ax            ;
[N84]  109:         mov   ax,205        ;═
[6C3]  110:         push  ax            ;
[5AB]  111:         jmp   z_end         ;zeichensatz 3 g
          eladen
[000]  112:
[2H6]  113: zeich4: mov   ax,214        ;╓
[6C3]  114:         push  ax            ;
[N54]  115:         mov   ax,183        ;╖
[6C3]  116:         push  ax            ;
[694]  117:         mov   ax,211        ;╙
[6C3]  118:         push  ax            ;
[B74]  119:         mov   ax,189        ;╜
[6C3]  120:         push  ax            ;
[G64]  121:         mov   ax,186        ;║
[6C3]  122:         push  ax            ;
[584]  123:         mov   ax,196        ;─
[91B]  124:         push  ax            ;zeichensatz 4 g
          eladen
```

```
[000]  125:
[WP7]  126: z_end:  ;Rahmen zeichnen
[7V7]  127:         ;linke untere Ecke zeichnen
[BN6]  128:         mov   dh,[bp+16]    ;zeile y2
[R07]  129:         mov   dl,[bp+22]    ;spalte x1
[31A]  130:         mov   bh,0          ;bildschirmseite
         bh #
[EX5]  131:         mov   ah,2          ;code-nr 2
[RM7]  132:         int   10h           ;cursor auf x1,y
         2
[8Y5]  133:         mov   cx,1          ;1 zeichen
[5C7]  134:         mov   al,[bp-6]     ;ascii '└'
[FD7]  135:         mov   bl,[bp+14]    ;attribut #
[ZY5]  136:         mov   ah,9          ;code-nr 9
[ZD8]  137:         int   10h           ;zeichen ausgebe
         n
[000]  138:
[OFB]  139:         ;untere horizontale Linie zeichnen
[90A]  140:         inc   dl            ;eine spalte wei
         ter
[EX5]  141:         mov   ah,2          ;code-nr 2
[E98]  142:         int   10h           ;cursor auf x1+1
         ,y2
[DM5]  143:         mov   ax,[bp+20]    ;ax=x2
[X96]  144:         sub   ax,[bp+22]    ;ax=ax-x1
[FY4]  145:         dec   ax            ;ax=ax-1
[7KA]  146:         mov   cx,ax         ;anzahl der '-'
         zeichen
[A27]  147:         mov   al,[bp-12]    ;ascii '-'
[KY5]  148:         mov   ah,9          ;code nr 9
[ZD8]  149:         int   10h           ;zeichen ausgebe
         n
[000]  150:
[6W7]  151:         ;recht untere Ecke zeichnen
[E07]  152:         mov   dl,[bp+20]    ;spalte x2
[EX5]  153:         mov   ah,2          ;code-nr 2
[SM7]  154:         int   10h           ;cursor auf x2,y
         2
[8Y5]  155:         mov   cx,1          ;1 zeichen
[NF7]  156:         mov   al,[bp-8]     ;ascii '┘'
[ZY5]  157:         mov   ah,9          ;code-nr 9
[ZD8]  158:         int   10h           ;zeichen ausgebe
         n
[000]  159:
[F7B]  160:         ;rechte vertikale Linie zeichnen
[GS8]  161:         dec   dh            ;eine zeile hoeh
         er
[EX5]  162:         mov   ah,2          ;code-nr 2
[Z48]  163:         int   10h           ;cursor auf x2,y
         2-1
[SZ6]  164:         mov   al,[bp-10]    ;ascii '│'
[GU7]  165: loop1:  mov   cx,1          ;1 zeichen
[ZY5]  166:         mov   ah,9          ;code-nr 9
[ZD8]  167:         int   10h           ;zeichen ausgebe
         n
[GS8]  168:         dec   dh            ;eine zeile hoeh
         er
[EX5]  169:         mov   ah,2          ;code-nr 2
[HT7]  170:         int   10h           ;cursor auf x2,d
         h
[24A]  171:         mov   ah,[bp+18]    ;vergleichswert
         y1
[LX4]  172:         cmp   ah,dh         ;dh-ah
[6G7]  173:         jnz   loop1         ;zf<>0:loop1
[000]  174:
[6A8]  175:         ;rechte obere Ecke zeichnen
[8Y5]  176:         mov   cx,1          ;1 zeichen
[NB7]  177:         mov   al,[bp-4]     ;ascii '┐'
[ZY5]  178:         mov   ah,9          ;code-nr 9
[ZD8]  179:         int   10h           ;zeichen ausgebe
         n
[000]  180:
[HBA]  181:         ;linke vertikale Linie zeichnen
[BN6]  182:         mov   dh,[bp+16]    ;zeile y2
[X46]  183:         dec   dh            ;zeile y2-1
[R07]  184:         mov   dl,[bp+22]    ;spalte x1
[EX5]  185:         mov   ah,2          ;code-nr 2
[Y48]  186:         int   10h           ;cursor auf x1,y
         2-1
[567]  187:         mov   al,[bp-10]     ;ascii '│'
[NU7]  188: loop2:  mov   cx,1          ;1 zeichen
[ZY5]  189:         mov   ah,9          ;code-nr 9
[ZD8]  190:         int   10h           ;zeichen ausgebe
         n
[TP8]  191:         dec   dh            ;zeilenregister
         -1
[EX5]  192:         mov   ah,2          ;code-nr 2
[GT7]  193:         int   10h           ;cursor auf x1,d
         h
[24A]  194:         mov   ah,[bp+18]    ;vergleichswert
         y1
[LX4]  195:         cmp   ah,dh         ;dh-ah
[GG7]  196:         jnz   loop2         ;zf<>0:loop2
[000]  197:
[3M7]  198:         ;linke obere Ecke zeichnen
[8Y5]  199:         mov   cx,1          ;1 zeichen
[NE7]  200:         mov   al,[bp-2]     ;ascii '┌'
[ZY5]  201:         mov   ah,9          ;code-nr 9
[ZD8]  202:         int   10h           ;zeichen ausgebe
         n
```

```
[000]  203:
[8XA]  204:           ;obere horizontale Linie zeichnen
[90A]  205:           inc   dl              ;eine spalte wei
         ter
[EX5]  206:           mov   ah,2            ;code-nr 2
[898]  207:           int   10h             ;cursor auf x1+1
         ,y1
[DM5]  208:           mov   ax,[bp+20]      ;ax=x2
[X96]  209:           sub   ax,[bp+22]      ;ax=ax-x1
[FY4]  210:           dec   ax              ;ax=ax-1
[1KA]  211:           mov   cx,ax           ;anzahl der '-'z
         eichen
[A27]  212:           mov   al,[bp-12]      ;ascii '-'
[ZY5]  213:           mov   ah,9            ;code-nr 9
[ZD8]  214:           int   10h             ;zeichen ausgebe
         n
[000]  215:
[VV9]  216:           ;stack aufräumen (Zeichensatz)
[BZ5]  217:           mov   cx,6            ;6 zeichen
[CP5]  218: del_st:   pop   ax              ;
[FGB]  219:           loop  del_st          ;bis 6 zeichen vo
         n stack
[000]  220:
[91C]  221:           ;benutzte Register wiederherstellen
[RY7]  222:           pop   bp              ;bp zurueckholen
[8RA]  223:           pop   dx              ;register zuruec
         kholen
[Z23]  224:           pop   cx              ;
[S23]  225:           pop   bx              ;
[H23]  226:           pop   ax              ;
[000]  227:
[67D]  228:           ;return und stack aufräumen (Parame
         ter)
[G97]  229:           ret   12              ;clean stack
[DS2]  230: frame1    ENDP
[GO2]  231: code      ENDS
[DDO]  232: END
```

Listing 1. Das Assembler-Quellprogramm der Rahmenroutine

```
[SU7]    0: 50 53 51 52 55 8B EC 8B 46 0C 3D 01 00 75 1B B8
[5Z7]    1: DA 00 50 B8 BF 00 50 B8 C0 00 50 B8 D9 00 50 B8
[3U7]    2: B3 00 50 B8 C4 00 50 EB 59 90 3D 02 00 75 1B B8
[YZ7]    3: C9 00 50 B8 BB 00 50 B8 C8 00 50 B8 BC 00 50 B8
[OY7]    4: BA 00 50 B8 CD 00 50 EB 39 90 3D 03 00 75 1B B8
[OY7]    5: D5 00 50 B8 B8 00 50 B8 D4 00 50 B8 BE 00 50 B8
[V18]    6: B3 00 50 B8 CD 00 50 EB 19 90 B8 D6 00 50 B8 B7
[YY7]    7: 00 50 B8 D3 00 50 B8 BD 00 50 B8 BA 00 50 B8 C4
[NN7]    8: 00 50 8A 76 10 8A 56 16 B7 00 B4 02 CD 10 B9 01
[S88]    9: 00 8A 46 FA 8A 5E 0E B4 09 CD 10 FE C2 B4 02 CD
[A18]   10: 10 8B 46 14 2B 46 16 48 8B C8 8A 46 F4 B4 09 CD
[VP7]   11: 10 8A 56 14 B4 02 CD 10 B9 01 00 8A 46 F8 B4 09
[XZ7]   12: CD 10 FE CE B4 02 CD 10 8A 46 F6 B9 01 00 B4 09
[958]   13: CD 10 FE CE B4 02 CD 10 8A 66 12 3A E6 75 EC B9
[858]   14: 01 00 8A 46 FC B4 09 CD 10 8A 76 10 FE CE 8A 56
[GV7]   15: 16 B4 02 CD 10 8A 46 F6 B9 01 00 B4 09 CD 10 FE
[KX7]   16: CE B4 02 CD 10 8A 66 12 3A E6 75 EC B9 01 00 8A
[T98]   17: 46 FE B4 09 CD 10 FE C2 B4 02 CD 10 8B 46 14 2B
[KX7]   18: 46 16 48 8B C8 8A 46 F4 B4 09 CD 10 B9 06 00 58
[BS4]   19: E2 FD 5D 5A 59 5B 58 C2 0C 00
```

Listing 2. Besitzen Sie keinen Assembler, so können Sie »frame1.bin« auch direkt eingeben

```
[N45]    0: program frame1;
[9EK]    1: procedure frame1(x1,x2,y1,y2,attr,typ:byte)
           ; external 'FRAME1.BIN';
[YL3]    2: var  i : byte;
[BH1]    3: begin
[NM2]    4:  clrscr;
[YV3]    5:  for i:=0 to 3 do
[BH1]    6:    begin
[2F2]    7:     delay(400);
[AZ6]    8:     frame1(i*4,40+i*4,i*4,10+i*4,10,i);
[TUO]    9:    end;
[5TO]   10: end.
```

Listing 3. Turbo Pascal-Programm zur Demonstration der vier Rahmentypen

Hires-Grafik mit 16 Farben am Schneider PC 1512

Im Normalfall stellt der Schneider PC 1512 im hochauflösenden Grafikmodus (640x200 Punkte) nur zwei Farben dar. Eine kleine Routine in Turbo Pascal bringt bis zu 16 Farben gleichzeitig auf den Bildschirm.

Der Schneider PC 1512 besitzt einen speziellen Grafikmodus, in dem er bei einer Auflösung von 640x200 Punkten 16 Farben darstellen kann. Dieser spezielle Grafikmodus wird durch Überlagerung von vier Bildschirmseiten mit jeweils unterschiedlichen Farben realisiert und vom BIOS unterstützt.

Wird nun auf zwei Bildschirmseiten derselbe Punkt in zwei unterschiedlichen Farben gesetzt, so erscheint am Bildschirm eine Mischfarbe dieser beiden Farben.

Das BIOS stellt dem Benutzer drei Grafik-Routinen zu Verfügung, die über den Interrupt 15hex aufgerufen werden. Die Funktion »3« legt dabei die Mischfarbe für weitere Grafikbefehle fest. Dabei werden gleichzeitig zwei Bildschirmseiten angesprochen. Diese Funktion ist in die Prozedur »color(Farbe)« eingebunden.

Die Funktion »4« legt die aktuelle Seite fest, von der gelesen werden soll. Sie liefert auch die Information über die Farbe eines Punkts. Dazu werden alle vier Bildschirmseiten geprüft und im Anschluß daran wird der Farbcode berechnet.

Diese Funktion steckt in der Prozedur »getpiccolor (x,y,Farbe)«.

Funktion »5« ordnet dem Bildschirmrahmen eine eigene, vom Grafikhintergrund verschiedene Farbe zu.

Die Prozeduren der Datei »hires16.inc« (Listing 1) erklären

Bild. 16 Farben sind im Hires-Modus des PC 1512 kein Problem

sich weitgehend selbst. Lediglich die Berechnung des Farbcodes benötigt eine nähere Erläuterung.

Als erstes überprüft die Prozedur »getpiccolor« auf allen vier Bildschirmseiten die Punktfarbe. Der Farbcode der sichtbaren Mischfarbe errechnet sich aus den vier Summanden. Dies sind immer Zweier-Potenzen der betreffenden Bildschirmseite, bei denen an der aktuellen Stelle ein Punkt gesetzt ist.

Sie dürfen mit der Erweiterung alle normalen Turbo Pascal-Grafikroutinen einsetzen, sofern diese die Farben nicht direkt beeinflussen (wie zum Beispiel »GraphBackground« oder »HiresColor«).

Wollen Sie mit der neuen Grafik-Routine beispielsweise eine gelbe Linie auf dem Bildschirm schreiben, rufen Sie als erstes die Prozedur »color« auf, um mit dem Parameter »Farbe« die aktuelle Zeichenfarbe auf gelb zu stellen.

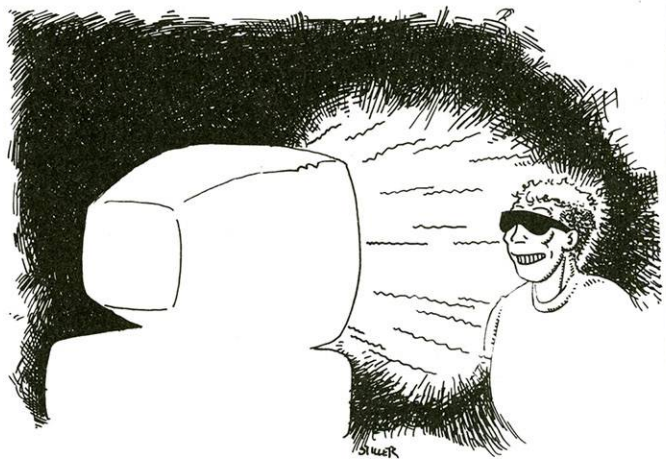
Dazu wird dem Parameter der Wert »14« (oder »yellow«) übergeben. (»yellow« ist eine vordefinierte Turbo Pascal-Konstante.)
Anschließend wird die Linie mit »draw (x1,y1,x2,y2,1)« gezeichnet.
Das Turbo Pascal-Programm »demo.pas« (Listing 2) demonstriert die Arbeitsweise der neuen Grafik-Routinen.

Nach dem Start sehen Sie Linien in 16 verschiedenen Farben (Bild). Um diese Farbvielfalt in anderen Programmen zu nutzen, binden Sie einfach die Include-Datei »hires16.inc« in die Software ein.

(Björn Bayard/sp)

```
[A21]    0: type
[9H5]    1:   Register = record
[S25]    2:                case integer of
[5LB]    3:                  1 : (AX,BX,CX,DX,BP,SI,DI,
           DS,ES,FLAGS : integer);
[H77]    4:                  2 : (AL,AH,BL,BH,CL,CH,DL,
           DH : byte);
[TUO]    5:              end;
[EMO]    6: var
[VU4]    7:   Regs     : Register;
[000]    8:
[NYM]    9: procedure color(Farbe : byte);          {
           Legt fest auf welche der vier }
[DGE]   10: begin                                   {
           Seiten geschrieben wird       }
[8W2]   11:   Regs.AH := 3;
[3K4]   12:   Regs.AL := Farbe;
[DY3]   13:   intr($15,Regs);
[TUO]   14: end;
[000]   15:
[ZZ9]   16: procedure border(Farbe : byte);
[BH1]   17: begin
[GW2]   18:   Regs.AH := 5;
[3K4]   19:   Regs.AL := Farbe;
[DY3]   20:   intr($15,Regs);
[TUO]   21: end;
[000]   22:
[RAK]   23: procedure getpiccolor(x,y : integer; var Fa
           rbe : integer);
[3R8]   24:
           { Liefert in "Farbe" die Farbe }
[197]   25:
           { des Punktes mit x,y          }
[EMO]   26: var
[KS7]   27:   Zaehler1, Zaehler2 : byte;
[FX6]   28:   _2_hoch_i          : integer;
[000]   29:
[69R]   30:   procedure aktuelle_Seite(Seite : byte);
           { Legt die Seite fest, von der  }
[DTC]   31:   begin
           { gelesen wird                  }
[CW2]   32:     Regs.AH := 4;
[LP4]   33:     Regs.AL := Seite;
[DY3]   34:     intr($15,Regs);
[TUO]   35:   end;
[000]   36:
[HVN]   37:   function Punkt(x,y : integer) : boolean;
           { Stellt fest ob der Punkt x,y }
[ZME]   38:   begin
           { auf der aktuellen Seite ge-  }
[SRE]   39:     Regs.AH := 13;
           { setzt ist oder nicht         }
[293]   40:     Regs.DX := y;
[S83]   41:     Regs.CX := x;
[DY3]   42:     intr($10,Regs);
[Z65]   43:     Punkt := (Regs.AL=1);
[TUO]   44:   end;
[000]   45:
[BH1]   46: begin
[6B2]   47:   Farbe := 0;
[ZDK]   48:   for Zaehler2 := 0 to 3 do
           { Alle vier Seiten durchtesten, }
[KME]   49:     begin
           { ob Punkt gesetzt oder nicht   }
[ZX7]   50:       aktuelle_Seite(Zaehler2);
[8HG]   51:       if Punkt(x,y)
           { Wenn Punkt gesetzt, dann ...  }
[B11]   52:         then
[BH1]   53:           begin
[OX3]   54:             Zaehler1 := 0;
[344]   55:             _2_hoch_i := 1;
[WP8]   56:             while (Zaehler1 < Zaehler2) do
[BH1]   57:               begin
[U88]   58:                 Zaehler1 := succ(Zaehler1);
[DY6]   59:                 _2_hoch_i := _2_hoch_i * 2
           ;
[TUO]   60:               end;
[NX3]   61:             if Zaehler1 = 0
[ZW4]   62:               then _2_hoch_i := 1;
[6TH]   63:             Farbe := Farbe + _2_hoch_i;
           { Ergibt Farbcode, der mit dem  }
[FVE]   64:           end;
           { von TURBO PASCAL uebereinstimmt}
[TUO]   65:     end;
[TUO]   66: end;
```

Listing 1. Das Listing der Include-Datei »hires.inc«



```
[U4B]    0: {-----------------------------------------
             ----------------------------
[P3M]    1:   Demo-Programm, dass 16 Farben bei einer A
           ufloesung von 640x200 Punkten
[G3M]    2:   auf dem Schneider PC 1512 zeigt. Benoetig
           t wird dafuer die Datei
[VLK]    3:   "HIRES16.INC", die die Grafikfunktionen v
           on TURBO-PASCAL um die Pro-
[BBL]    4:   zeduren "color(Farbe)", "getpiccolor(x,y,
           Farbe)" und "border(Farbe)"
[U43]    5:   erweitert.
[000]    6:
[R6L]    7:   ACHTUNG: Dieses Programm laeuft nur auf d
           em Schneider PC 1512 !!!
[000]    8:
[BDF]    9:   Autor: Björn Bayard / v. Humboldtstr. 80
           / 5024 Pulheim
[HZA]   10:   -----------------------------------------
             ---------------------------- }
[000]   11:
[G84]   12: program demo;
[000]   13:
[ZM3]   14: {$I HIRES16.INC}
[000]   15:
[EMO]   16: var
[8H5]   17:   i,Farbe : integer;
[000]   18:
[BH1]   19: begin
[5Y1]   20:   hires;
[654]   21:   for i := 1 to 16 do
[BH1]   22:     begin
[5M2]   23:       color(i);
[234]   24:       draw(1,200,i*40,1,1);
[TUO]   25:     end;
[654]   26:   for i := 1 to 16 do
[BH1]   27:     begin
[5M2]   28:       color(i);
[VZ4]   29:       draw(640,200,640-i*40,1,1);
[TUO]   30:     end;
[654]   31:   for i := 1 to 16 do
[BH1]   32:     begin
[2M2]   33:       border(i);
[HM2]   34:       delay(1000);
[TUO]   35:     end;
[1B2]   36:   color(12);
[253]   37:   plot(100,100,1);
[107]   38:   getpiccolor(100,100,Farbe);
[BV7]   39:   repeat until keypressed;
[813]   40:   textmode;
[V25]   41:   writeln(Farbe);
[5TO]   42: end.
```

Listing 2. Der Quellcode des Demonstrationsprogrammes »demo.pas«

DOS-Commander macht MS-DOS einfach

Eine Benutzeroberfläche macht auch die Eingabe kompliziertester MS-DOS-Anweisungen einfach. Die Utility steht dabei immer im Hintergrund und ist auch nach einem Programmaufruf aktiv. Besonders das Durchsuchen verschiedener Unterdirectories ist mit diesem Hilfsprogramm eine Kleinigkeit.

Der DOS-Commander ist eine Hilfe für die tägliche Arbeit mit MS-DOS. Die wichtigsten Anweisungen des Betriebssystems werden menügesteuert auf einen Tastendruck hin aktiviert. Kopieren, Löschen, Umbenennen – all diese Funktionen können Sie mit dem DOS-Commander aufrufen, ohne die ausführliche Syntax des Befehls zu kennen.
Nach dem Start des Commanders wird auf dem Bildschirm das Startbild ausgegeben (siehe Bild). Es ist in insgesamt fünf Bereiche aufgeteilt:

- das Optionsmenü
- das Eingangsverzeichnis (linkes Fenster)
- das aktive Unterverzeichnis (rechtes Fenster)
- der Kommandobereich
- der Bereich für Hinweise

Am Anfang steht der Lichtbalken, der die aktive Version anzeigt, in der Optionszeile. Mit »DIR« werden in dem linken mittleren Fenster die Dateien des aktuellen Directories angezeigt. Jetzt befindet sich der Lichtbalken über der ersten Datei dieses Verzeichnisses. Mit den Cursortasten können Sie ihn auf eine andere Datei stellen. Drücken Sie [Return], so wird das markierte Programm aufgerufen und gestartet. Sie können natürlich nur lauffähige Programme starten, also Dateien mit der Namenserweiterung ».bat«, ».com« oder ».exe«. Alle anderen Startversuche enden mit einer Fehlermeldung des Betriebssystems. Ist ein so aktiviertes Programm zu Ende ausgeführt, so wird der DOS-Commander automatisch wieder aufgerufen. Mit der Esc-Taste verlassen Sie das Fenster, und es ist wieder an Ihnen, andere Funktionen aus dem Optionsmenü aufzurufen.
Drücken Sie die Return-Taste, während der Lichtbalken auf dem Namen eines Unterverzeichnisses steht, so wird dieses aktiviert. Die Dateien erscheinen im rechten Fenster. Mit der Datei ».« positionieren Sie die Anzeige im linken Fenster; dort können Sie auf die anderen Funktionen aus dem Optionsmenü zugreifen. Das rechte Fenster wird dabei gelöscht.
Mit der Datei »..« wird das nächst höhere Verzeichnis aufgerufen – wie bei dem MS-DOS-Befehl »cd ..«. Mit diesen beiden Punkt-Dateien lassen sich innerhalb kürzester Zeit auch größte Datenbestände durchforsten.
Um nun eine Datei zu bearbeiten (umbenennen, ausdrucken, kopieren und so weiter), müssen Sie sie markieren. Dabei hilft die F1-Taste, mit der Sie die Datei, auf der der Lichtbalken steht, als »markiert« kennzeichnen. Sie können aber nur Dateien in dem linken Fenster markieren, denn dieses Fenster dient speziell der Bearbeitung von Dateien.
Hierbei wird in der Hinweiszeile die Meldung

```
Markierung läuft !!!
```

sowie der Name der entsprechenden Datei ausgegeben.
Nachdem Sie die zu behandelnden Dateien so markiert haben, beenden Sie die Routine mit [F2] (Funktion beenden) oder [F3] (Abbrechen).

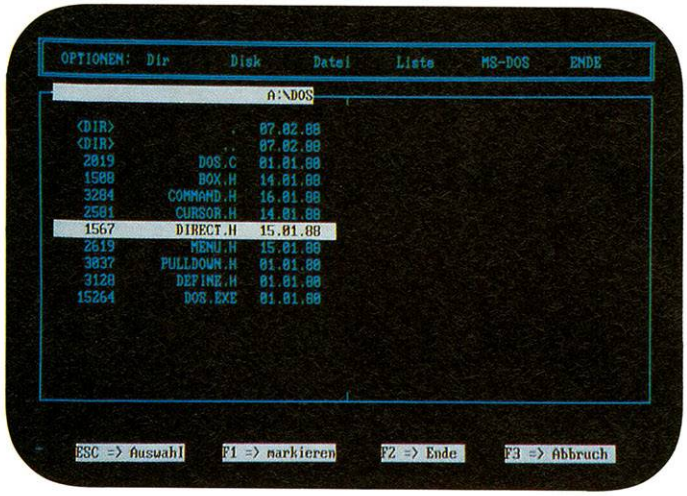


Bild. Das Hauptmenü des DOS-Commanders

Der Auswahlbalken steht wieder in dem Optionsmenü und es steht Ihnen frei, wieder andere Funktionen aufzurufen. Es sind aber nur die Menüpunkte »DATEI« und »LISTE« interessant.
Der Menüpunkt »LISTE« gibt alle markierten Dateien der Reihe nach auf dem Bildschirm, auf einem Drucker oder in eine Datei aus. Haben Sie versehentlich eine falsche Datei markiert, so können Sie sie in der Kommandozeile mit »?« wieder deaktivieren. In der Kommandozeile werden auch alle anderen Informationen abgefragt – beispielsweise der Name der Zieldatei oder der neue Name beim Umbenennen. Sämtliche Eingaben müssen hierbei den Konventionen von MS-DOS entsprechen. Benötigen Sie ein Kommando oder eine Datei häufiger, so wiederholt ein Druck auf die F3-Taste – wie unter MS-DOS – die letzte Eingabe. Überhaupt ist die Anlehnung an MS-DOS jederzeit zu spüren. So kann mit der Funktion »MS-DOS« auch jede Anweisung des Betriebssystems mit Buchstaben eingegeben werden.
Bei der Option »Datei löschen« ist die Markierfähigkeit allerdings eingeschränkt. Es läßt sich nur eine einzige Datei markieren. Und noch einen Punkt müssen Sie bei der Arbeit mit dem DOS-Commander beachten: Während der Markierfunktion dürfen Sie das Verzeichnis nicht wechseln, denn dabei werden alle Markierungen zurückgesetzt. In der Version aus Listing 1 können Sie bis zu zehn Dateien im aktuellen Verzeichnis markieren. Dieses Limit ist jedoch beliebig zu verändern.
Bevor Sie den Programmcode aus Listing 1 compilieren, geben Sie bitte die Dateien aus den Listings 2 bis 8 ein und speichern sie unter dem angegebenen Namen. Die wichtigste Include-Datei des Programms ist »define.h«. Sie beherbergt sämtliche Strukturen, globale Variablen sowie verschiedene Preprozessor-Anweisungen. Von besonderer Bedeutung ist dabei die Reihenfolge der einzelnen Anweisungen. Weichen Sie keinesfalls davon ab, da das Programm sonst nicht korrekt läuft. Die Anpassung an eine Hercules- oder CGA-Grafikkarte erfolgt ebenfalls in diesem Programmteil. In den Kommentarzeilen finden Sie die entsprechenden Anweisungen. Die abgedruckte Version ist für die Hercules-Karte zuständig.
Der DOS-Commander läuft unter Turbo C. Die Optionen in »Less Common Errors« (Options – Compiler – Errors – Less Common Errors) müssen alle auf »off« gestellt werden. Für Quick C (und auch viele andere C-Compiler) sind die beiden Funktionen »peekb« und »pokeb« neu zu schreiben. Diese Funktionen erlauben das Lesen, beziehungsweise das Schreiben an eine bestimmte Speicherstelle. Weitere Änderungen sind zumindest für Quick C nicht nötig.

(Harald Krause/ma/hg)

```
[8LA]    0: /*****************************************
            ******************************/
[42D]    1: /*                    DOS-Commander Version 1
            .0                              */
[6UC]    2: /*                    Autor: Harald Krause
                                            */
[OKA]    3: /*                           Geigerstr. 2
                                            */
[XPA]    4: /*                           8500 Nürnberg 60
                                            */
[8LA]    5: /*****************************************
            ******************************/
[000]    6:
[7Z5]    7: #include <define.h>
[000]    8:
[OF1]    9: main()
[V30]   10: {
[XF4]   11:  int i=1,j=0,subi;
[733]   12:  cursoff();
[W12]   13:  maske();
[D72]   14:  while(1){
[044]   15:    clsxy(22,0,23,79,7);
[VB7]   16:    i=pulldown(1,13,0,6,8,3,8,0,i,-1);
[L14]   17:    subi=menue(i);
[XX2]   18:    switch(i) {
[GL5]   19:      case 1: kill_seite();
[X06]   20:              dirload();dir_aus();
[D72]   21:              while(1){
[SLE]   22:                 j=pulldown(5,spalte1[cont],
            1,16,36,0,1,0,1,anz[cont]);
[4D5]   23:                 if(!j || j==-2) break;
[UXD]   24:                 if(!strcmp(root[cont][j].da
            t_groesse,"<DIR>"))
[RW3]   25:                    konv_komm(j);
[73F]   26:                  else {strcpy(kommando,root
            [cont][j].dat_name);
[C54]   27:                    cont=0;do_dos();
[X30]   28:                 }
[F33]   29:               dir_aus();
[X30]   30:              }
[KV1]   31:              break;
[0U5]   32:      case 2:  if(subi!=4) {
[2V3]   33:               switch(subi){
[D7A]   34:                 case 1: accept_kom("LAUFWER
            K(mit :)>");
[NX7]   35:                         strcpy(kommando,"")
            ; break;
[3G9]   36:                 case 2: accept_kom("Neues S
            UBDIR>");
[NGA]   37:                         strcpy(kommando,"mk
            dir ");break;
[F1A]   38:                 case 3: accept_kom("Lösche
            SUBDIR>");
[5HA]   39:                         strcpy(kommando,"rm
            dir ");break;
[X30]   40:               }
[VL8]   41:               strcat(kommando,hilf);cont=0
            ;
[ZG2]   42:               do_dos();
[X30]   43:              }
[KV1]   44:              break;
[000]   45:
[C49]   46:      case 3: if(subi!=4 && mergecount)
[B15]   47:                 do_merge(subi-1);
[KV1]   48:              break;
[000]   49:
[K49]   50:      case 4: if(subi!=4 && mergecount)
[615]   51:                 do_merge(subi+2);
[KV1]   52:              break;
[000]   53:
[468]   54:      case 5:accept_kom("KOMMANDO>");
[HXC]   55:             strcpy(kommando,hilf);cont=0;do_
            dos();break;
[000]   56:
[UL6]   57:      case 6:curson();clrscr();
[7V1]   58:             exit(0);
[X30]   59:      }
[X30]   60:    }
[X30]   61:  }
```

Listing 1. Der DOS-Commander macht die Arbeit unter MS-DOS einfach

```
[MU9]    0: /*****************************************
            ************************/
[HMG]    1: /* box.h --> enthält Routinen zum Maskenauf
            bau                     */
[MU9]    2: /*****************************************
            ************************/
[49D]    3: box(azeile,aspalte,bzeile,bspalte,form)
[VBD]    4: int azeile,aspalte,bzeile,bspalte,form;
[V30]    5: {
[45F]    6:    unsigned char ecke1,ecke2,ecke3,ecke4,qu
            er,laengs;
[GM1]    7:    int i;
[KW2]    8:    if(form){
[Y2G]    9:          ecke1=218;ecke2=191;ecke3=192;ecke4
            =217;quer=196;laengs=179;}
[HM1]   10:       else{
[81G]   11:          ecke1=201;ecke2=187;ecke3=200;ecke
            4=188;quer=205;laengs=186;}
[RL8]   12:    xchar(azeile,aspalte,ecke1,1);
[AM8]   13:    xchar(bzeile,bspalte,ecke4,1);
[ZL8]   14:    xchar(azeile,bspalte,ecke2,1);
[2M8]   15:    xchar(bzeile,aspalte,ecke3,1);
[90E]   16:    xchar(azeile,aspalte+1,quer,bspalte-aspa
            lte-1);
[GOE]   17:    xchar(bzeile,aspalte+1,quer,bspalte-aspa
            lte-1);
[XS9]   18:    for(i=1;i<=(bzeile-azeile-1);i++)   {
[1A9]   19:       xchar(azeile+i,aspalte,laengs,1);
[VH9]   20:       xchar(azeile+i,bspalte,laengs,1);}
[X30]   21: }
[XAA]   22: /*****************************************
            ************************/
[20G]   23: /* linie --> zieht Linie im angebenen Modus
                                            */
[000]   24:
[01B]   25: linie(azeile,aspalte,laenge,zeichen)
[V78]   26: int azeile,aspalte,laenge;
[X87]   27: unsigned char zeichen;
[V30]   28: {
[09B]   29:    xchar(azeile,aspalte,zeichen,laenge);
[X30]   30: }
[OCA]   31: /*****************************************
            ************************/
[EUE]   32: /* senk --> zieht senkrechte Linie
                                            */
[000]   33:
[B68]   34: senk(zeile,spalte,anzahl)
[SC8]   35:   int zeile,spalte,anzahl;
[V30]   36: {
[ZP2]   37:   int i,abso;
[PB6]   38:   abso=zeile*160+spalte*2;
[S68]   39:   for(i=0;i<anzahl;i++,abso+=160)
[NG5]   40:      pokeb(0xb000,abso,179);
[X30]   41: }
```

Listing 2. »box.h« enthält Routinen für den Maskenaufbau

```
[GE9]    0: /*****************************************
            ************************/
[2BN]    1: /* command.h --> stellt Routinen zur Komman
            dobearbeitung bereit  */
[GE9]    2: /*****************************************
            ************************/
[000]    3:
[DA9]    4: /*****************************************
            ************************/
[9KH]    5: /* konv_komm --> prüft ob "cd ." -Kommando
            erfolgen soll         */
[000]    6:
[DL3]    7: konv_komm(i)
[GM1]    8: int i;
[V30]    9: {
[7Y6]   10:   strcpy(kommando,"cd ");
[Y1D]   11:   strcat(kommando,root[cont][i].dat_name);
[EK8]   12:   if(!strcmp(kommando,"cd ."){
[BZ4]   13:      wechsel_seite();
[RP2]   14:      return;
[X30]   15:   }
[F22]   16:   cont=1;
[2S3]   17:   kill_seite();
[3G5]   18:   system(kommando);
[LF4]   19:   dirload(cont);
[1A4]   20:   verz_aus(3,42);
[X30]   21: }
[000]   22:
[N2A]   23: /*****************************************
            ************************/
[UWL]   24: /* wechsel_seite --> veranlasst den Austaus
            ch der Windows          */
[000]   25:
[BZ4]   26: wechsel_seite()
[V30]   27: {
[GM1]   28:   int i;
[AB5]   29:   if(!cont) return;
[B57]   30:   for(i=1;i<=anz[cont];i++){
[7AF]   31:      strcpy(root[0][i].dat_name,root[1][i].d
            at_name);
[H7H]   32:      strcpy(root[0][i].dat_groesse,root[1][i
            ].dat_groesse);
[MED]   33:      root[0][i].dat_datum=root[1][i].dat_dat
            um;
[X30]   34:   }
[AT7]   35:   anz[0]=anz[1];kill_seite();
[F78]   36:   anz[1]=0;cont=0;kill_seite();
[2L9]   37:   gotoxy(3,2);printf("%34s",akt_ver);
[X30]   38: }
[000]   39:
[OCA]   40: /*****************************************
            ************************/
[BZH]   41: /* kill_seite --> löscht die aktuelle Windo
```

```
          wseite                          */
[000]     42:
[VE3]     43: kill_seite()
[V30]     44: {
[DEC]     45:     clsxy(5,spalte1[cont],20,spalte2[cont],7
          );
[X30]     46: }
[000]     47:
[XAA]     48: /*****************************************
          ****************************/
[WNN]     49: /* dir_aus --> gibt das aktuelle Directory
          in der aktuellen Seite aus */
[000]     50:
[TZ2]     51: dir_aus()
[V30]     52: {
[A3H]     53:   dirprint(5,spalte1[cont],(anz[cont]>15)?1
          6:anz[cont],1);
[X30]     54: }
[000]     55:
[XAA]     56: /*****************************************
          ****************************/
[BPH]     57: /* verz_aus --> gibt das aktuelle Verzeichn
          is aus                          */
[000]     58:
[Z04]     59: verz_aus(x,y)
[GE2]     60:  int x,y;
[V30]     61: {
[873]     62:  int i,start;
[RL4]     63:  start=x*160+y*2+1;
[C56]     64:  akt_ver=getcwd(0,80);
[1R6]     65:  if(strlen(akt_ver)>33)
[3CC]     66:     strcpy(akt_ver,"   Verzeichnis zu lang")
          ;
[893]     67:  gotoxy(x,y);
[8K6]     68:  printf("%34s",akt_ver);
[VDC]     69:  for(i=start;i<start+67;i+=2) pokeb(SCREEN,
          i,112);
[X30]     70: }
[000]     71:
[CCA]     72: /*****************************************
          ****************************/
[4MH]     73: /* do_dos --> führt das gewünschte DOS-Komm
          ando aus                        */
[000]     74:
[4F2]     75: do_dos()
[V30]     76: {
[K34]     77:  char *strchr();
[A87]     78:  if(!strchr(kommando,'?')){
[KN5]     79:   clrscr();curson();
[3G5]     80:   system(kommando);
[8LL]     81:   writestr("Zurück zum DOS-Commander mit be
          liebiger Taste",112);
[G12]     82:   getch();
[NU5]     83:   cursoff();maske();
[X30]     84:  }
[X30]     85: }
[5FA]     86: /*****************************************
          ****************************/
[VSK]     87: /* accept_kom --> nimmt Kommando für do_dos
          () entgegen                     */
[000]     88:
[285]     89: accept_kom(out)
[DD2]     90: char *out;
[V30]     91: {
[9C3]     92:  gotoxy(23,0);
[748]     93:  writestr(out,112);curson();
[TY2]     94:  gets(hilf);
[733]     95:  cursoff();
[X30]     96: }
[CCA]     97: /*****************************************
          ****************************/
[LOM]     98: /* do_merge --> bereitet zusammengesetztes
          DOS-Kommando vor                */
[000]     99:
[7M3]    100: do_merge(nr)
[GC2]    101:  int nr;
[V30]    102: {
[7Y2]    103:  int count;
[CNC]    104:  for(count=0;count<mergecount;count++) {
[044]    105:     clsxy(22,0,23,79,7);
[8C3]    106:     gotoxy(22,0);
[ZRK]    107:     printf("Datei: %s",root[cont][merge_tab
          [count]].dat_name);
[LW5]    108:     if(!nr) mergecount=0;
[9C3]    109:     gotoxy(23,0);
[587]    110:     accept_kom(tab[nr][0]);
[DV8]    111:     strcpy(kommando,tab[nr][1]);
[XFK]    112:     strcat(kommando,root[cont][merge_tab[co
          unt]].dat_name);
[016]    113:     strcat(kommando," ");
[6RB]    114:     if(nr==5) strcat(kommando,tab[nr][2]);
[KY6]    115:     strcat(kommando,hilf);
[YHB]    116:     if(nr!=5) strcat(kommando,tab[nr][2]);
[ZG2]    117:     do_dos();
[X30]    118:   }
[HV3]    119:   mergecount=0;
[X30]    120: }
```

Listing 3. In »command.h« stehen die Kommandos

```
[569]     0: /*****************************************
          ****************************/
[XVK]     1: /* cursor.h --> stellt Bildschirmsteuerrout
          inen bereit          */
[000]     2:
[3P5]     3: union REGS Register;
[000]     4:
[244]     5: void clrscr(x)
[VP1]     6: int x;
[V30]     7: {
[NE3]     8:   if (x!=112) x=7;
[UY4]     9:   Register.h.ah = 6;
[8Z4]    10:   Register.h.al = 0;
[015]    11:   Register.h.bh = x;
[C25]    12:   Register.x.cx = 0;
[545]    13:   Register.h.dh = 24;
[855]    14:   Register.h.dl = 79;
[VD8]    15:   int86(0x10,&Register,&Register);
[X30]    16: }
[X07]    17: /*****************************************
          ******/
[VK6]    18: void clsxy(x1,y1,x2,y2,x)
[A44]    19: int x1,y1,x2,y2,x;
[V30]    20: {
[NE3]    21:   if (x!=112) x=7;
[UY4]    22:   Register.h.ah = 6;
[8Z4]    23:   Register.h.al = 0;
[015]    24:   Register.h.bh = x;
[165]    25:   Register.h.ch = x1;
[P65]    26:   Register.h.cl = y1;
[765]    27:   Register.h.dh = x2;
[W65]    28:   Register.h.dl = y2;
[VD8]    29:   int86(0x10,&Register,&Register);
[X30]    30: }
[057]    31: /*****************************************
          *******/
[B25]    32: void gotoxy(x,y)
[GE2]    33: int x,y;
[V30]    34: {
[PY4]    35:   Register.h.ah = 2;
[SY4]    36:   Register.h.bh = 0;
[815]    37:   Register.h.dh = x;
[X15]    38:   Register.h.dl = y;
[VD8]    39:   int86(0x10,&Register,&Register);
[X30]    40: }
[4H7]    41: /*****************************************
          *********/
[VE6]    42: void write(zeichen,x)
[VP1]    43: int x;
[YW3]    44: char zeichen;
[V30]    45: {
[9V4]    46:  Register.h.ah =9;
[NN7]    47:  Register.h.al = zeichen;
[M15]    48:  Register.h.bl = x;
[D25]    49:  Register.x.cx = 1;
[SY4]    50:  Register.h.bh = 0;
[VD8]    51:  int86(0x10,&Register,&Register);
[X30]    52: }
[G28]    53: /*****************************************
          ***********/
[N88]    54: void writestr(string,farbe)
[YU2]    55: int farbe;
[GH3]    56: char *string;
[V30]    57: {
[2R4]    58:     while(*string){
[AD4]    59:     write('',farbe);
[S35]    60:     Register.h.ah = 14;
[SY4]    61:     Register.h.bh = 0;
[9W7]    62:     Register.h.al = *string++;
[VD8]    63:     int86(0x10,&Register,&Register);
[X30]    64:     }
[X30]    65:  }
[000]    66:
[000]    67:
[G28]    68: /*****************************************
          ***********/
[UGD]    69: void xchar(zeile,spalte,zeichen,anzahl)
[SC8]    70: int zeile,spalte,anzahl;
[X87]    71: unsigned char zeichen;
[V30]    72: {
[YC6]    73:   gotoxy(zeile,spalte);
[3Y4]    74:   Register.h.ah=10;
[5W4]    75:   Register.h.bh=0;
[VM6]    76:   Register.x.cx=anzahl;
[Y87]    77:   Register.h.al=zeichen;
[VD8]    78:   int86(0x10,&Register,&Register);
[X30]    79: }
[G28]    80: /*****************************************
          ***********/
[DP3]    81: int getcurs()
[V30]    82: {
[GE2]    83:   int x,y;
[MY4]    84:   Register.h.ah = 0;
[7E8]    85:   int86(0x16,&Register,&Register);
[D55]    86:   x=Register.h.al;
[K45]    87:   y=Register.h.ah;
[W62]    88:   if(!x){
[2S2]    89:      switch(y){
```

```
[FB5]   90:        case 59: return(-1);
[S95]   91:        case 60: return(-2);
[1A5]   92:        case 61: return(-3);
[845]   93:        case 72: return(1);
[145]   94:        case 80: return(2);
[Z45]   95:        case 75: return(3);
[E55]   96:        case 77: return(4);
[2S6]   97:        default: return(0);}
[X30]   98:        }
[HR2]   99:        return(x);
[X30]  100: }
[5F8]  101: /*****************************************
           ****************/
[6P3]  102: int cursoff()
[V30]  103: {
[CU4]  104:  Register.h.ah= 1;
[OY4]  105:  Register.h.ch= -1;
[MY4]  106:  Register.h.cl= -1;
[VD8]  107:  int86(0x10,&Register,&Register);
[X30]  108: }
[XX8]  109: /*****************************************
           ******************/
[LA3]  110: int curson()
[V30]  111: {
[CU4]  112:  Register.h.ah= 1;
[MU4]  113:  Register.h.ch= 0;
[8V4]  114:  Register.h.cl= 7;
[VD8]  115:  int86(0x10,&Register,&Register);
[X30]  116: }
```

Listing 4. »cursor.h« stellt Routinen für die Bildschirmsteuerung bereit

```
[XL9]    0: /*****************************************
            ************************/
[2OM]    1: /* define.h  --> enthält alle Preprozessor-
            Anweisungen sowie die  */
[7VE]    2: /*                 benötigten statischen Vari
            ablen                  */
[XL9]    3: /*****************************************
            ************************/
[000]    4:
[GVG]    5: #define SCREEN 0xb000            /*Hercules-
            Grafikkarte;CGA=0xb800*/
[RVG]    6: #define DIR_MAX 512              /*maximale
            Directory-Einträge     */
[V7G]    7: #define MERGE_MAX 10             /*maximale
            Markierungen           */
[000]    8:
[52G]    9: static int cont;                 /*Aktuelle
            Windowseite            */
[VVL]   10: static int spalte1[2]={2,42};    /*Startpunk
            t Window (oben links)  */
[V2M]   11: static int spalte2[2]={39,77};   /*Endpunkt
            Window (oben rechts)   */
[S1K]   12: static int anz[2];               /*Anzahl de
            r Directory-Einträge   */
[000]   13:
[GKH]   14: struct verz {                    /*eigene St
            ruktur für Directory   */
[MO5]   15:         char dat_name[15];
[U46]   16:         char dat_groesse[8];
[OL4]   17:         int dat_datum;
[4F5]   18: } root[2][DIR_MAX];
[000]   19:
[S5H]   20: struct dat {                     /*Struktur
             zur Datum-Umwandlung  */
[PX4]   21:         int tag,mon,ja;
[MUO]   22: }da;
[000]   23:
[GPG]   24: static struct windowstr{        /*Window-S
            truktur                */
[U32]   25:          int   pos;
[RM8]   26:          unsigned char  inhalt[15][50];
[X33]   27:          int   breite;
[653]   28:          int   laenge;
[VW3]   29: } win_tab[2];
[000]   30:
[XCK]   31: static men[5][8]=               /*Tabellen
             zum Aufbau von Bildschirm-*/
[6P9]   32:       {1, 3, 1, 8,0, 0, 2,79,    /*masken
                                      */
[063]   33:        0, 0, 0, 0,0, 0, 0, 0,
[LK3]   34:        4,24, 8,12,2,21, 9,37,
[MN3]   35:        4,35,12,16,2,32, 9,48,
[5L4]   36:        4,46,16,20,2,43, 9,59};
[000]   37:
[6F5]   38: static char *menu[]=
[XVC]   39:  {" ","OPTIONEN:","Dir","Disk","Datei","Lis
            te","MS-DOS","ENDE",
[VC9]   40:   "LW-Wechsel","mkdir","rmdir","ABBRUCH",
[LKC]   41:   "löschen","umbenennen","kopieren","ABBRUC
            H",
[KPA]   42:   "Bildschirm","Drucker","Datei","ABBRUCH"}
           ;
[000]   43:
[4ZD]   44: static char *tab[6][3]={"LÖSCHEN (Stop=?)>"
            ,"del ","",
[FD9]   45:                        "UMBENENNEN (neuer
            Name)>","ren ","",
[VA9]   46:                        "KOPIEREN (neuer Na
            me)>","copy ","",
[65B]   47:                        "LISTE Bildschirm(S
            top=?)>","type ",">con",
[EHA]   48:                        "LISTE Drucker(Stop
            =?)>","type ",">prn",
[G87]   49:                        "LISTE in DAT (Name
            )>","type ",">"};
[000]   50:
[ABL]   51: static char *akt_ver;          /*Hilfsvariab
            le zur "system"-Funktion */
[000]   52:
[X1E]   53: static mergecount;             /*Markierhilf
            en                     */
[L38]   54: static merge_tab[MERGE_MAX];
[000]   55:
[UDK]   56: #include <stdio.h>               /*Standarde
            in- und Ausgaberoutinen*/
[UZD]   57: #include <dos.h>                /*MS-DOS-Ro
            utinen                 */
[E4F]   58: #include <dir.h>                /*Directory
            -Routinen              */
[F6L]   59: #include <cursor.h>            /*Routinen zu
            r Bildschirmsteuerung*/
[RDE]   60: #include <box.h>               /*Rahmen-Rout
            inen                   */
[VGE]   61: #include <menu.h>              /*Window-Rout
            inen                   */
[FNH]   62: #include <pulldown.h>          /*Steuerung d
            es Pulldown-Menue      */
[000]   63:
[50G]   64: struct ffblk dir;              /*definierte
            Struktur in DIR.H      */
[000]   65:
[8GH]   66: #include <direct.h>            /*Besondere D
            irectory-Routinen      */
[000]   67:
[SXF]   68: char kommando[80];             /*Buffer-Vari
            able für main()        */
[FNB]   69: char hilf[60];                 /*Hilfsfeld
                                   */
[000]   70:
[9KG]   71: #include <command.h>           /*Hilfsroutin
            en für main()          */
```

Listing 5. Sämtliche Definitionen stehen in »define.h«

```
[GE9]    0: /*****************************************
            ************************/
[CZL]    1: /* direct.h --> stellt Routinen zum Laden v
            on Directories bereit */
[GE9]    2: /*****************************************
            ************************/
[000]    3:
[GE9]    4: /*****************************************
            ************************/
[SKG]    5: /* dirload --> lädt aktuelles Directory in
            Tabelle                */
[000]    6:
[124]    7: dirload(mode)
[LP2]    8: int mode;
[V30]    9: {
[KK2]   10:   int ok,i=1;
[TH7]   11:   ok=findfirst("*.*",&dir,255);
[NV3]   12:   while(ok>-1) {
[YZK]   13:       if(dir.ff_attrib==8 || dir.ff_attrib=
            =20 || dir.ff_attrib==40);
[CX1]   14:       else {
[EAF]   15:         ltoa(dir.ff_fsize,root[mode][i].dat
            _groesse,10);
[LBD]   16:         strcpy(root[mode][i].dat_name,dir.f
            f_name);
[V4C]   17:         root[mode][i].dat_datum=dir.ff_fdat
            e;
[EKC]   18:         if(dir.ff_attrib==16 || dir.ff_attr
            ib==48)
[VVC]   19:         strcpy(root[mode][i].dat_groesse,"<
            DIR>");
[CHO]   20:         i++;
[X30]   21:       }
[P65]   22:    ok=findnext(&dir);
[X30]   23:   }
[F84]   24:   anz[mode]=--i;
[X30]   25: }
[000]   26:
[MU9]   27: /*****************************************
            ************************/
[R2H]   28: /* dirprint --> gibt gewünschtes Directory
```

```
          aus                                   */
[000]   29:
[KH7]   30: dirprint(ze,sp,anza,nr)
[WV4]   31: int ze,sp,anza,nr;
[V30]   32: {
[GM1]   33:   int i;
[WS5]   34:   for(i=1;i<=anza;i++){
[DM4]   35:     gotoxy(ze++,sp);
[XSP]   36:     printf(" %7s    %12s",root[cont][nr].da
          t_groesse,root[cont][nr].dat_name);
[UKA]   37:     datum(root[cont][nr].dat_datum);
[58C]   38:     printf("   %02d.%02d.%02d",da.tag,da.mo
          n,da.ja);
[U51]   39:     ++nr;
[X30]   40:   }
[X30]   41: }
[000]   42:
[DA9]   43: /*****************************************
          **********************/
[SCK]   44: /* datum --> wandelt einen Integer-Wert in
          Datei-Datum um        */
[000]   45:
[X33]   46: datum(wert)
[LU4]   47: unsigned wert;
[V30]   48: {
[WB4]   49:   da.mon=wert>>5;
[TL2]   50:   da.mon&=15;
[RV3]   51:   da.ja=wert>>9;
[G42]   52:   da.ja+=80;
[T34]   53:   da.tag=wert&31;
[X30]   54: }
```

Listing 6. Mit »direct.h« werden Directories geladen

```
[569]    0: /*****************************************
           ********************/
[4SG]    1: /* menu.h --> enthält Routinen zur Window-S
           teuerung          */
[569]    2: /*****************************************
           ********************/
[000]    3:
[DEF]    4: /* winsave --> speichert angegenes Window a
           b                 */
[000]    5:
[KPD]    6: int winsave(wnr,zeile,spalte,breite,laenge)
[11B]    7:     int wnr,zeile,spalte,breite,laenge;
[V30]    8: {
[A74]    9:   int start,i,j;
[9M2]   10:   breite*=2;
[SO7]   11:   start=zeile*160+spalte*2;
[T58]   12:   win_tab[wnr].pos  =start;
[GL8]   13:   win_tab[wnr].breite=breite;
[ZE8]   14:   win_tab[wnr].laenge=laenge;
[SK8]   15:   for(i=0;i<laenge;i++,start+=160)
[YL5]   16:       for(j=0;j<breite;j++)
[NRL]   17:         win_tab[wnr].inhalt[i][j]=(unsigned
          char)peekb(SCREEN,start+j);
[HR2]   18:   return(0);
[000]   19:
[X30]   20: }
[000]   21:
[569]   22: /*****************************************
           ********************/
[00G]   23: /* winload --> baut angebenes Window wieder
           auf               */
[000]   24:
[EF5]   25: int winload(wnr)
[6T2]   26:     int wnr;
[V30]   27: {
[W82]   28: int i,j;
[FRG]   29:  for(i=0;i<win_tab[wnr].laenge;i++,win_tab[
          wnr].pos+=160)
[53A]   30:       for(j=0;j<win_tab[wnr].breite;j++)
[T6F]   31:             pokeb(SCREEN,win_tab[wnr].pos+j,
          (unsigned char)
[VE8]   32:                   win_tab[wnr].inhalt[i][j])
          ;
[HR2]   33:  return(0);
[X30]   34: }
[000]   35:
[569]   36: /*****************************************
           ********************/
[T9F]   37: /*menue --> erstellt gewüschtes Menue
                             */
[000]   38:
[PT3]   39: int menue(nr)
[GC2]   40:     int nr;
[V30]   41: {
[W82]   42:  int i,j;
[SO3]   43:  switch(nr){
[6T4]   44:     case 0: box(0,0,2,79,0);
[UCC]   45:             for(i=men[nr][2],j=0;i<men[nr][
          3];i++,j+=11){
[SZ9]   46:               gotoxy(men[nr][0],men[nr][1]+j)
          ;
[2Z6]   47:               writestr(menu[i],7);}
[EV1]   48:     case 1:
[6W1]   49:     case 5:
[XL4]   50:     case 6: return(0);
[000]   51:
[45M]   52:     default:winsave(0,men[nr][4],men[nr][5]
          ,men[nr][7]-men[nr][5],
[G77]   53:                     men[nr][6]-men[nr][4]);
[000]   54:
[ROK]   55:             clsxy(men[nr][4]+1,men[nr][5]+1
          ,men[nr][6]-1,men[nr][7]-1,112);
[BUB]   56:             for(i=men[nr][2],j=0;i<men[nr][
          3];i++,j++){
[BY9]   57:                 gotoxy(men[nr][0]+j,men[nr]
          [1]);
[8W6]   58:                 writestr(menu[i],112);
[X30]   59:             }
[UEG]   60:             box(men[nr][4]+1,men[nr][5]+1,m
          en[nr][6]-1,men[nr][7]-1,1);
[AKH]   61:             j=pulldown(men[nr][0],men[nr][1
          ],1,men[nr][3]-men[nr][2],10,
[NS1]   62:                        0,1,1,1,-1);
[353]   63:             winload(0);
[X30]   64:   }
[HR2]   65:   return(j);
[X30]   66: }
[000]   67:
[NSA]   68: /*****************************************
           ********************************/
[DWF]   69: /* maske --> baut die Eingangsmaske auf
                                          */
[000]   70:
[W12]   71: maske()
[V30]   72: {
[GM1]   73:  int i;
[BK2]   74:  clrscr();
[962]   75:  menue(0);
[A23]   76:  box(3,0,21,79,1);
[3R7]   77:  gotoxy(3,40);writestr("┬",7);
[O78]   78:  gotoxy(21,40);writestr("┴",7);
[DW2]   79:  senk(4,40,17);
[61B]   80:  gotoxy(24,5);writestr("ESC => Auswahl",112
          );
[FSC]   81:  gotoxy(24,24);writestr("F1 => markieren",1
          12);
[82A]   82:  gotoxy(24,45);writestr("F2 => Ende ",112);
[H7B]   83:  gotoxy(24,61);writestr("F3 => Abbruch ",11
          2);
[LF4]   84:  dirload(cont);
[F33]   85:  dir_aus();
[VY3]   86:  verz_aus(3,2);
[X30]   87: }
[000]   88:
```

Listing 7. »menu.h« ist für das Menü zuständig

```
[DA9]    0: /*****************************************
           ********************/
[CPM]    1: /* pulldown.h --> ermöglicht die Programmie
           rung eines P-Menue's */
[DA9]    2: /*****************************************
           ********************/
[000]    3:
[P9F]    4: pulldown(ze,sp,ab,max,br,le,anzb,mode,st,to
           tal)
[PFC]    5: int ze,sp,ab,max,br,le,anzb,mode,st,total;
[V30]    6: {
[RO9]    7:  int hell,norm,j,i,l,akt=1,s=0;
[D52]    8:  char c;
[SZ4]    9:  unsigned tab[61];
[CW7]   10:  if(total>-1) mergecount=0;
[4B7]   11:  ze*=160;sp*=2;br*=2;le*=2;ab*=160;
[7CB]   12:  if((total<max)&&(total>-1)) max=total;
[PY7]   13:  if (!mode) { hell=112;norm=7;}
[417]   14:  else {hell=7;norm=112;}
[EH5]   15:  for(i=1,j=0;i<=max;j++)
[ZX5]   16:      for(l=0;l<anzb;l++,i++)
[A79]   17:          tab[i]=ze+(ab*j)+sp+(le*l)+(br*l)+
           1;
[M38]   18:  inver(tab[i=st],br,hell);
[SD6]   19:  while((c=getcurs())!=13){
[AC3]   20:     gotoxy(24,0);
[3T6]   21:     inver(tab[i],br,norm);
[2S2]   22:     switch(c){
[2G5]   23:     case 1:if(max>anzb) {
[HS3]   24:                if(i-anzb>1) {
[ZW4]   25:                    i-=anzb; --akt;}
[WE2]   26:                else { i=1;
[8D6]   27:                   if((akt-1<=1)&&(total>-1))
           {
[YE1]   28:                       akt=1;
[AZ1]   29:                       if(s==1){
[N14]   30:                           scroll(1,akt);
[U61]   31:                           s=0;}
[X30]   32:                       }
```

```
[KM9]   33:                    else if(total>-1) scroll(1
        ,--akt);
[X30]   34:                 }
[X30]   35:               }
[KV1]   36:                break;
[D85]   37:        case 2:if(max>anzb){
[7A4]   38:               if(i+anzb>max) {
[8V1]   39:                    i=max;
[1S8]   40:                    if((akt+1<=total)&&(total>-
        1)) {
[CLO]   41:                       s=1;
[TR4]   42:                       scroll(0,++akt);}
[925]   43:                    else akt=total;
[X30]   44:                 }
[CX1]   45:                 else {
[OD4]   46:                    i+=anzb;++akt;
[X30]   47:                 }
[X30]   48:               }
[KV1]   49:                break;
[XLC]   50:        case 4: if(anzb>1) i=(i+1>=max)?max:++i
        ;break;
[CMA]   51:        case 3: if(anzb>1) i=(i-1<=1)?1:--i;bre
        ak;
[A88]   52:        case 27:if(total>-1) return(0);
[HE7]   53:        case -1:if(total>-1 && cont==0) {
[KAD]   54:                    gotoxy(22,0);writestr("MARKI
        ERUNG läuft!!!",112);
[W6D]   55:                    gotoxy(22,23);printf("Mark-N
        r:%2d",mergecount+1);
[BG7]   56:                    if(mergecount<MERGE_MAX){
[OC5]   57:                      dirprint(23,0,1,akt);
[5H8]   58:                      merge_tab[mergecount]=akt;
[X24]   59:                      ++mergecount;
[X30]   60:                    }
[X30]   61:                  }
[KV1]   62:                  break;
[8E8]   63:        case -2:if(total>-1) return(c);
[XV5]   64:        case -3:if(total>-1) {
[044]   65:                  clsxy(22,0,23,79,7);
[HV3]   66:                  mergecount=0;
[X30]   67:                }
[KV1]   68:                break;
```

```
[FW6]   69:       default:printf("%c",7);
[X30]   70:       }
[LR6]   71:     inver(tab[i],br,hell);
[X30]   72:    }
[3T6]   73:   inver(tab[i],br,norm);
[ZB7]   74:   return((total>-1)?akt:i);
[X30]   75:   }
[DA9]   76: /*****************************************
        *********************/
[AYF]   77: /* inver --> ändert das Bidschirmattribut
                                                  */
[000]   78:
[C57]   79: inver(akt,breite,wert)
[M16]   80: int akt,breite,wert;
[V30]   81: {
[GM1]   82:   int i;
[RLD]   83:   for(i=0;i<=breite;i+=2) pokeb(SCREEN,akt+
        i,wert);
[X30]   84: }
[000]   85:
[GE9]   86: /*****************************************
        *********************/
[Y2H]   87: /* scroll --> ermöglicht das Scrolling eine
        s Windows                 */
[000]   88:
[R75]   89: scroll(modus,nr)
[WW3]   90: int modus,nr;
[V30]   91: {
[T03]   92: int x=20,y=6;
[TH5]   93: if(modus) {y=7;x=5;}
[OC5]   94:   Register.h.ah=y;
[WW4]   95:   Register.h.al=1;
[PX4]   96:   Register.h.bh=7;
[CX4]   97:   Register.h.ch=5;
[OP8]   98:   Register.h.cl=spalte1[cont];
[PY4]   99:   Register.h.dh=20;
[9P8]  100:   Register.h.dl=spalte2[cont];
[VD8]  101: int86(0x10,&Register,&Register);
[PDA]  102: dirprint(x,spalte1[cont],1,nr);
[X30]  103: }
```

Listing 8. Pulldown-Menüs mit »pulldown.h«

Struktur gefällig?

Übersicht ist Trumpf – zumindest bei Programmen. Die Utility »Rpltab« leistet zum Schreiben strukturierter Quelltexte gute Dienste: Das C-Programm hilft Ihnen nämlich beim Überarbeiten von Tabulatoren in Ihren Programmen.

Bei der Entwicklung von C- oder Pascal-Programmen kommt der optischen Gestaltung des Quellcodes eine große Bedeutung zu. Eine klare Struktur verbessert die Lesbarkeit eines Programmtexts entscheidend. Im wesentlichen wird dabei mit der Einrückung der einzelnen Programmzeilen entsprechend der hierarchischen Struktur des Programms gearbeitet. Das typische Aussehen solch strukturierter Texte ist jedem Softwareentwickler bekannt.

Um Programme schnell einzugeben, bedient man sich beim Einrücken der Programmzeilen häufig der Tabulatoren. Diese liefern mit wenig Aufwand eine saubere optische Struktur. Die Wirkung der Tabulator-Taste ist jedoch von Editor zu Editor verschieden. Der eine unterstützt die Tabulator-Taste, indem er bis zur nächsten Tabulator-Position Leerzeichen einfügt. In der internen Textdarstellung wird aber der ASCII-Wert 9 beibehalten.

Beim Editieren solch einer Datei bemerkten Sie das Vorhandensein eines Tabulators erst dann, wenn mit der Leertaste beim Überschreiben eine ganze Reihe von Leerzeichen gelöscht werden.

Andere Editoren füllen auch in der internen Textdarstellung sofort Leerstellen ein. Zu dieser Gattung zählt auch der Editor von Turbo Pascal. Bei gut strukturierten Programmtexten führt diese Art der Textdarstellung leicht zu »aufgeblähten« Quell-Dateien.

Wollen Sie mit solch einem Editor ASCII-Texte bearbeiten, die echte Tabulatoren (ASCII-Wert 9) enthalten, so bleibt von der ursprünglichen Struktur nicht mehr viel übrig. Die Tabulatoren werden nicht durch die entsprechende Zahl von Leerzeichen, sondern durch ein inverses »I« dargestellt. Es gibt noch eine ganze Reihe weiterer Fälle, bei denen es notwendig ist, Leerzeichen in Tabulatoren und Tabulatoren in Leerzeichen zu verwandeln.

Der Algorithmus für die Umwandlung von Tabulatoren in Leerzeichen und umgekehrt ist relativ einfach. Zur Umwandlung von Tabulatoren in Leerzeichen muß die Tabulator-Schrittweite und die aktuelle Zeichenposition gegeben sein. Die Zahl der benötigten Leerstellen (i), die durch einen Tabulator zu ersetzen sind, errechnet sich aus folgender Formel:

```
i = Schrittweite – Rest von (aktuelle Zeichenposition /
 Schrittweite)
```

Ist zum Beispiel das vierte Zeichen einer Textzeile ein Tabulator, so müssen bei einer Schrittweite von acht Zeichen (Tabulator-Positionen 1, 9, 17, ...) vier Leerzeichen eingefügt werden:

```
i = 8 - Rest (4/8) = 4
```

Sind Tabulatoren mit der Schrittweite zehn gesetzt und ist das zwölfte Zeichen einer Zeile ein Tabulator, so sind acht Leerzeichen einzufügen:

```
i = 10 - Rest (12/10) = 8
```

Bei der Umwandlung von Leerzeichen in Tabulatoren wird eine Textzeile in Teilstrings von maximal »s« Zeichen zerlegt, wobei »s« die Schrittweite angibt, mit der die Tabulator-Positionen gesetzt werden sollen. Bei allen diesen Teilstrings werden die zusammenhängenden, am Ende des Teilstrings stehenden Leerzeichen gelöscht und durch einen Tabulator ersetzt. Die Zahl der zu löschenden Leerstellen wird durch zeichenweises Lesen des Teilstrings von hinten nach vorne ermittelt.

»Rpltab« ist vollständig in C geschrieben, um auch längere Quellcodes rasch bearbeiten zu können. Das Programm läßt sich mit Turbo C, Microsoft C (ab Version 3.0) oder

Quick C compilieren. Der Aufruf des jeweiligen Compilers ist im Programmkopf erläutert.
Die Funktion »main()« testet nacheinander alle beim Programmaufruf angegebenen Parameter. In dieser Routine sind außer »Optionen« keine weiteren Argumente zugelassen. Optionen können mit » – « oder »/« eingeleitet werden. Das Programm hält sich damit sowohl an den Unix- als auch den MS-DOS-Standard. Die Optionen dürfen Sie sowohl in Klein- als auch in Großbuchstaben angeben. Registriert das Programm ein unzulässiges Argument oder eine fehlerhafte Option, bricht es mit einer Fehlermeldung ab. Der Exit-Code des Programms ist in solch einem Fall größer Null.
Nachdem alle Programmparameter gesetzt sind, wird die Funktion »work()« aufgerufen, die den eigentlichen Kern dieser Utility bildet. Im ersten Teil von »work()« wird eine Textzeile zeichenweise bis zum Zeilenende eingelesen. Findet die Routine dabei ein ASCII-Zeichen mit dem Wert 9 (Tabulator), so ersetzt sie dieses durch die nötige Anzahl von Leerstellen. Dabei verwendet das Programm die Variablen »InTabPos«, um die Schrittweite des Tabulators zu speichern und »i« als Zähler für die Zahl der gelesen Zeichen (einschließlich der ersetzten Leerzeichen). Ist im zweiten Teil von »work()« die Option »-S« gesetzt, so wird die zwischengespeicherte Textzeile sofort ausgegeben. Im anderen Fall ersetzt Rpltab soweit als möglich Leerstellen durch Tabulatoren. Die Umwandlung der Leerzeichen in Tabulatoren erfolgt ab der Programmzeile 229. In diesem Programmteil werden Teilstrings der Eingabezeile gebildet, die höchstens so lang wie die Tabulator-Schrittweite sind. Die letzten Leerzeichen jeder dieser Zeichenketten werden nacheinander durch die Zeichen »\t« und »\0« überschrieben.

Dieses C-Programm hilft nicht nur beim Schreiben übersichtlicher und strukturierter Programme – es ist auch selbst vorbildlich programmiert

C-Programme sind sehr oft unübersichtlich und schwer zu durchschauen. Dies liegt in erster Linie daran, daß C eine sehr komplexe Formulierung von Programmanweisungen erlaubt. Es ist nicht unbedingt böswillige Absicht des Programmierers, unlesbare Quell-Codes zu schreiben. Schließlich soll mit dieser komplexen Formulierung der Anweisungen ein möglichst effizientes Programm entstehen. Um so mehr muß der Programmierer mit einem geeigneten Programmlayout und einer ausführlichen Dokumentierung seines Programms für die Lesbarkeit sorgen. Die Utility Rpltab dient selbst als Beispiel für einen besonders durchdachten und strukturierten Quellcode. Wesentliche Merkmale sind die Gliederung des Programmtextes durch grafische Elemente wie Linien und Kästen sowie eine einheitliche Schreibweise von dokumentierten Programmzeilen.
Nahezu jedes C-Programm läßt sich grob in folgende Abschnitte gliedern:

- generelle Programminformationen (Verwendung, Compilerhinweise)
- Copyrightvermerk
- Include-Files, Konstanten und Makros
- Externe Variablen und Funktionen
- Funktionsübersicht
- diverse Funktionen

Diese Abschnitte wurden bei »Rpltab« mit auffälligen Kästen umrahmt, beziehungsweise mit einer großen Überschrift versehen (Listing). Damit fällt sofort die entsprechende Funktion des jeweiligen Programmteils ins Auge. Ein Großteil der notwendigen Programmdokumentation erledigt sich alleine mit dem Schreiben des Programmtexts und dem Einhalten des vereinbarten Programm-Layouts. Aus Gründen der Übersichtlichkeit traf man eine besondere Regelung bei der Schreibweise von Kommentaren zu einzelnen Programmzeilen:
Alle Kommentare beginnen mit der Zeichenfolge »/*« auf Spalte 41 einer Textzeile. Ferner werden alle Kommentare rechtsbündig mit Leerstellen aufgefüllt und natürlich mit der Zeichenfolge »*/« abgeschloßen. Diese Regel bewirkt, daß die C-Anweisungen selbst im linken Teil des Texts nicht von Kommentaren unterbrochen werden und klar lesbar sind. Die Arbeit mit C hat gezeigt, daß Kommentare, die unmittelbar in die Anweisungsfolge geschrieben werden, häufig nicht mehr von den reinen Anweisungen zu unterscheiden sind.
Die vollständige Syntax der Funktion lautet:

```
rpltab [-?| -H] [-S] [-In] [-On] [<Datei1] [>Datei2]
```

Alle Optionen können Sie alternativ mit » – « oder »/« einleiten und sowohl Groß- als auch Kleinbuchstaben benutzen. Dem Benutzer stehen die folgende Optionen zur Auswahl:

`-?` oder `-H`

Wird eine diese Optionen gewählt, gibt das Programm eine Kurzbeschreibung aus und kehrt zur MS-DOS-Ebene zurück.

-S

Diese Option bewirkt, daß bei der Ausgabe die Leerzeichen nicht durch Tabulatoren ersetzt werden. Dies ist nützlich, wenn das Ausgabegerät (Drucker) oder der Texteditor (zum Beispiel Turbo Pascal), mit dem Sie die Datei bearbeiten wollen, keine Tabulatoren verarbeiten kann.

-In

Beim Lesen der Datei wird nicht mit der Tabulator-Schrittweite 8, sondern mit Schrittweite »n« gearbeitet. Jede n-te Position ist somit eine Tabulatorposition. Zwischen »-I« und »n« darf kein Leerzeichen stehen. Eine richtige Eingabe ist zum Beispiel »-I16«. Die Option »-I0« hat die gleiche Wirkung wie »-I1«: Jeder Tabulator wird durch eine Leerstelle ersetzt.

-On

»-On« arbeitet wie die Option »-In«, bezieht sich jedoch auf die Ausgabe der Datei. Diese Option ist wirkungslos, wenn »-S« aktiviert wurde.
Zum besseren Verständnis abschließend noch vier typische Beispiele für den Einsatz von Rpltab.
Beispiel 1:

```
rpltab > text1
```

Die Tastatureingaben werden in die Datei »text1« geschrieben und mit [Ctrl-Z] beendet. In der Datei »text1« werden Leerstellen soweit wie möglich durch Tabulatoren ersetzt. Dieser Aufruf entspricht folgender Standardvorgabe:

```
rpltab -i8 -o8 >text1
```

Beispiel 2:

```
rpltab -o4 <text1 >text2
```

Die Datei »text1« wird eingelesen, wobei alle Tabulatoren mit der Schrittweite 8 durch Leerzeichen ersetzt werden. Die Ausgabe erfolgt in die Datei »text2«. Dabei werden Tabulatoren mit der Schrittweite 4 gesetzt.
Beispiel 3:

```
rpltab -s <source1.pas >source2.pas
```

In der Datei »source1.pas« werden alle Tabulatoren in Leerstellen umgewandelt. Bei der Ausgabe in die Datei »source2.pas« werden keine neuen Tabulatoren gesetzt. Sie können »source2.pas« dann zum Beispiel mit dem Turbo Pascal-Editor problemlos bearbeiten.
Beispiel 4:

```
rpltab <source2.pas >source1.pas
```

Umwandeln einer Textdatei, die mit dem Editor von Turbo Pascal geschrieben wurde, in ein Textfile mit Tabulato-

ren (zur Platzersparnis). Sie können diesen Aufruf auch benutzen, um die Tabulatoren einer mehrfach bearbeiteten Textdatei (unregelmäßiges Einfügen und Löschen von Tabulatoren und Leerzeichen) zu bereinigen. Sie dürfen bei den Optionen »-In« und »-On« grundsätzlich nur positive Schrittweiten angeben, denn negative Werte führen zu nicht vorhersehbarem Programmablauf. Weiterhin verarbeitet Rpltab nur Textzeilen mit einer maximalen Länge von 255 Zeichen. Treten während der Bearbeitung längere Zeilen auf, bricht das Programm mit einer Fehlermeldung ab. Die Zahl der Zeichen bezieht sich auf die Länge der Textzeile inklusive Tabulator-Sprünge.

(Manfred Beer/sp)

```
[9GE]    0: /***** Programm rpltab.c ******************
           ************************************/
[9V8]    1: /*
                                                */
[OLM]    2: /*  Programm zur Handhbung von Tabulatoren
           in Textdateien:                      */
[GAK]    3: /*          - Ersetzen von Tabulatoren durch
           Leerzeichen                          */
[EEK]    4: /*          - Ersetzen von Leerzeichen durch
           Tabulatoren                          */
[9V8]    5: /*
                                                */
[Y7E]    6: /*  Betriebssystem :  MS-DOS ab 2.00
                                                */
[9V8]    7: /*
                                                */
[95E]    8: /*  Compiler        :  MICROSOFT C ab Vers.
           3.00                                 */
[9V8]    9: /*
                                                */
[M2F]   10: /*          Compilieren.................: >m
           sc rpltab;                           */
[Z4E]   11: /*          Linken......................: >l
           ink rpltab;                          */
[NEA]   12: /*          oder
                                                */
[T3G]   13: /*          Compilieren und Linken .....: >c
           l rpltab;                            */
[9V8]   14: /*
                                                */
[69D]   15: /*  Compiler        :  TURBO C ab Vers. 1.00
                                                */
[9V8]   16: /*
                                                */
[XHH]   17: /*          Compilieren und Linken .....: >t
           cc -mt -w- -f rpltab;                */
[FCN]   18: /*          ggf. umwandeln in .com-File.: >e
           xe2bin rpltab.exe rpltab.com         */
[9V8]   19: /*
                                                */
[TUF]   20: /*          Compiler-Optionen...........: -m
           t  Model ... Tiny                    */
[FVD]   21: /*                                   -w
           -  Display warnings ... off          */
[4GD]   22: /*                                   -f
           -  Floating paoint ... none          */
[9V8]   23: /*
                                                */
[5CM]   24: /*  Programmaufruf : >rpltab [-?!-H] [-S] [
           -In] [-On]  [I/O-Umleitung]          */
[ENC]   25: /*                  (s.u.: Funktion Hilfe(
           ))                                   */
[9V8]   26: /*
                                                */
[NFB]   27: /****************************************
           ************************************/
[F9P]   28: /* Copyright (C)   : Ingenieurbuero Beer -
           Hauptstrasse 69 - 7505 Ettlingen 3 */
[NFB]   29: /****************************************
           ************************************/
[9V8]   30: /*
                                                */
[CKB]   31: /*                   R E V I S I O N   H I
           S T O R Y                            */
[9V8]   32: /*
                                                */
[8GF]   33: /* Datum    ! Erlaeuterung
                          !Vers.! von */
[D8C]   34: /*----------+-----------------------------
           ---------------------+-----+-----*/
[WPE]   35: /* 87-06-08 ! erster Programmentwurf
                          !  00 ! MB  */
[W2G]   36: /* 87-08-21 ! endgueltiges Programm-Layout
                          !  01 ! MB  */
[KW8]   37: /*           !
                          !     !     */
[NFB]   38: /****************************************
           ************************************/
[000]   39:
[B35]   40:                                      /*
           <f>                                  */
[000]   41:
[NFB]   42: /****************************************
           ************************************/
```

```
[ZRF]   43: /*  I. Include-Files, Konstanten und Makros
                                                */
[NFB]   44: /****************************************
           ************************************/
[000]   45:
[YK7]   46:                                      /*
           Include-Files                        */
[YW7]   47:                                      /*
           ================================== */
[SU5]   48: # include <stdio.h>
[M1G]   49: # include <stdlib.h>                  /*
           nur fuer LINT_ARGS                   */
[27D]   50: # include <string.h>                  /*
           ...                                  */
[000]   51:
[C57]   52:                                      /*
           Konstanten                           */
[YW7]   53:                                      /*
           ================================== */
[F85]   54: # define MAXCHAR  255
[6X4]   55: # define SPACE     32
[1W4]   56: # define TAB      '\t'
[HP5]   57: # define NEWLINE '\n'
[VF4]   58: # define YES         1
[K94]   59: # define NO          0
[N3K]   60: # define OPT_ERR    1                 /*
           fehlerhaft Option angegeben          */
[N7K]   61: # define CMD_ERR    2                 /*
           Fehler in der Kommandozeile          */
[TZF]   62: # define LEN_ERR    3                 /*
           zu lange Textzeile                   */
[000]   63:
[A96]   64:                                      /*
           Defaults                             */
[YW7]   65:                                      /*
           ================================== */
[HEK]   66: # define INTABPOS   8                 /*
           TAB-Schrittweite beim Einlesen       */
[1MK]   67: # define OUTTABPOS  8                 /*
           TAB-Schrittweite bei der Ausgabe     */
[000]   68:
[000]   69:
[NFB]   70: /****************************************
           ************************************/
[4NG]   71: /*  II. Externe Variablen und Funktionen
                                                */
[NFB]   72: /****************************************
           ************************************/
[000]   73:
[BLC]   74: void  exit();                         /*
           s. stdlib.h                          */
[2ND]   75: char *strncpy();                      /*
           s. string.h                          */
[49D]   76: int   toupper();                      /*
           s. stdlib.h                          */
[8Y2]   77: unsigned
[K2D]   78: int   strlen();                       /*
           s. string.h                          */
[000]   79:
[000]   80:
[NFB]   81: /****************************************
           ************************************/
[2XG]   82: /*  III. Interne Variablen und Funktionen
                                                */
[NFB]   83: /****************************************
           ************************************/
[000]   84:
[VS2]   85: int  main();
[M53]   86: void work();
[1M3]   87: void Fehler();
[163]   88: void Hilfe();
[000]   89:
[Z66]   90: int  InTabPos  = INTABPOS;
[KX6]   91: int  OutTabPos = OUTTABPOS;
[7W4]   92: int  stripTab  = NO;
[0B3]   93: int  Line      = 1;
[000]   94:
[NFB]   95: /****************************************
           ************************************/
[K3E]   96: /*  IV. Funktionsuebersicht
                                                */
[NFB]   97: /****************************************
           ************************************/
[000]   98:
```

```
[000]   99:
[NFB]  100: /*****************************************
       **********************************/
[LYD]  101: /* Name              ! Beschreibung
                                      !mode*/
[H8C]  102: /*------------------+----------------------
       ------------------------------+----*/
[85M]  103: /* main()            ! rpltab Main-Funktion,
        Auswertung Kommandozeile     !    */
[6XK]  104: /* work()            ! rpltab Arbeitsfunktio
       n, TAB's bearbeiten          !    */
[KEK]  105: /* Fehler()          ! Hinweis auf Benutzung
       sfehler anzeigen             !    */
[EME]  106: /* Hilfe()           ! online Programmhilfe
                                    !    */
[NFB]  107: /*****************************************
       **********************************/
[000]  108:
[B35]  109:                                          /*
       <f>                                           */
[000]  110:
[Z8C]  111: /*---------------------------------------
       ----------------------------------*/
[YFC]  112: /* function    :   main()
                                                     */
[Z8C]  113: /*---------------------------------------
       ----------------------------------*/
[T3H]  114: /* description : <c> mafarbld Startroutine
                                                     */
[PSM]  115: /*               - rpltab MAin-Funktion, Au
       swertung der Kommandozeile                    */
[DXB]  116: /*               - Aufruf work()
                                                     */
[Z8C]  117: /*---------------------------------------
       ----------------------------------*/
[000]  118:
[LRC]  119: int main (argc, argv)
                                                     /*
[HUA]  120: ====================
                                                     */
[YN2]  121: int argc;
[EL2]  122: char **argv;
[V30]  123: {
[VZ3]  124:     char *cmdlarg;
[000]  125:
[FBN]  126:     while (cmdlarg = *++argv)      /* alle
       Kommandozeilen Parameter auswerten */
[V30]  127:     {
[GAF]  128:         if (*cmdlarg == '-' || *cmdlarg ==
       '/') /* Option ?                          */
[C6M]  129:         switch (toupper(*++cmdlarg))    /*
       Option normieren und verzw.               */
[V30]  130:         {
[DWA]  131:             case '?' :                  /*
       Option -H oder -?                         */
[812]  132:             case 'H' :
[HSC]  133:                 Hilfe();                /*
       Prommhilfe ausgeben                       */
[18D]  134:                 exit(0);                /*
       und Programm abbrechen                    */
[000]  135:
[6X9]  136:             case 'S' :                  /*
       Option -s                                 */
[8EG]  137:                 stripTab = YES;         /*
       alle Tabulatoren entfernen                */
[KV1]  138:                 break;
[000]  139:
[OV9]  140:             case 'I' :                  /*
        Option -I                                */
[TZ8]  141:                 sscanf(cmdlarg+1, "%d", &In
       TabPos);
[87D]  142:                                         /* TAB-
       Schrittweite beim Einlesen waehlen */
[SM3]  143:                 if (!InTabPos)
[H43]  144:                     InTabPos =1;
[KV1]  145:                 break;
[000]  146:
[AW9]  147:             case 'O' :                  /*
       Option -O                                 */
[RG9]  148:                 sscanf(cmdlarg+1, "%d", &Ou
       tTabPos);
[SGD]  149:                                         /* TAB-Sc
       hrittweite bei der Ausgabe waehlen */
[ZA4]  150:                 if (!OutTabPos)
[H43]  151:                     InTabPos =1;
[KV1]  152:                 break;
[000]  153:
[LKD]  154:             default  :                  /*
       unbekannte Option                         */
[FTG]  155:                 Fehler(OPT_ERR, cmdlarg); /
       * Fehlermeldung ausgeben                   */
[VCD]  156:                 exit (OPT_ERR);         /*
       Programm abbrechen                        */
[000]  157:
[X30]  158:         }
```

```
[R9C]  159:         else                            /*
       keine Option                              */
[V30]  160:         {
[BXF]  161:                                     /* nur Option
       en in der Kommandozeile zugelassen */
[S2H]  162:             Fehler (CMD_ERR, cmdlarg);  /*
       Fehlermeldung ausgeben                    */
[BLD]  163:             exit (CMD_ERR);             /*
       Programm abbrechen                        */
[X30]  164:         }
[X30]  165:     }
[RMC]  166:     work();                             /*
       Text bearbeiten                           */
[MSD]  167:     return (0);                         /*
       Exitcode setzen                           */
[2DB]  168: }                                       /*
       end of main()                             */
[Z8C]  169: /*---------------------------------------
       ----------------------------------*/
[000]  170:
[045]  171:                                         /*
       <f> 15                                    */
[000]  172:
[Z8C]  173: /*---------------------------------------
       ----------------------------------*/
[BAC]  174: /* Funktion     : work()
                                                  */
[Z8C]  175: /*---------------------------------------
       ----------------------------------*/
[AHH]  176: /* Beschreibung : <c> rpltab-Arbeitsroutine
                                                  */
[C3C]  177: /*                - TAB's bearbeiten
                                                  */
[Z8C]  178: /*---------------------------------------
       ----------------------------------*/
[000]  179:
[HRA]  180: void work()
                                                  /*
[OT9]  181: ===========
                                                  */
[V30]  182: {
[ZW5]  183:     char Input[MAXCHAR+1];
[8M6]  184:     char Output[MAXCHAR+1];
[622]  185:     int c, i=0;
[000]  186:
[000]  187:
[9GB]  188:     /*-----------------------------------
       ----------------------------------*/
[LDP]  189:     /* stdin zeichenweise einlesen bis eing
       elesenes Zeichen EOF ist                  */
[9GB]  190:     /*-----------------------------------
       ----------------------------------*/
[TYH]  191:     while ((c = getchar()) != EOF)      /*
       Datei zeichenweise einlesen               */
[V30]  192:     {
[FSE]  193:         if (i >= MAXCHAR)               /*
       Ueberlauf Textbuffer !                    */
[V30]  194:         {
[EU4]  195:             Fehler(LEN_ERR, "");
[9H3]  196:             exit(LEN_ERR);
[X30]  197:         }
[000]  198:
[S7F]  199:         if (c != NEWLINE)               /*
       noch kein Ende Textzeile                  */
[V30]  200:         {
[X8A]  201:             /*-----------------------------
       ----------------------------------*/
[MMK]  202:             /* wenn eingelesenes Zeichen TA
       B, dann Zeichen in SPACEs umw.            */
[X8A]  203:             /*-----------------------------
       ----------------------------------*/
[TTA]  204:             if (c == TAB)               /*
       TAB gefunden                              */
[V30]  205:             {
[SO9]  206:                                         /*
       TAB's durch SAPCE ersetzen                */
[W06]  207:                 while ((i+1)%InTabPos != 0)
[OBD]  208:                     Input[i++] = SPACE; /*
       SPACE in Textbuffer                       */
[GY1]  209:                 c=SPACE;
[X30]  210:             }
[BBK]  211:             Input[i++]=(char) c;        /*
       eingelesenes Zeichen in Textbuffer */
[X30]  212:         }
[ABE]  213:         else                            /*
       Ende Textzeile gefunden                   */
[V30]  214:         {
[X8A]  215:             /*-----------------------------
       ----------------------------------*/
[LEM]  216:             /* Textzeile auf stdout ausgebe
       n, SPACEs ggf. in TABs umwandeln          */
[X8A]  217:             /*-----------------------------
       ----------------------------------*/
[KO4]  218:             Input[i]='\0';
[625]  219:             if (stripTab == YES)
```

```
[48A]  220:                                          /*
       Ausgabe ohne Tabulatoren                 */
[WS5]  221:                    printf("%s\n", Input);
[A11]  222:                else
[V30]  223:                {
[7U9]  224:                    /*-------------------------
       ------------------------------------*/
[DHB]  225:                    /* SPACEs in TABs umwandeln
                                            */
[7U9]  226:                    /*-------------------------
       ------------------------------------*/
[U26]  227:                    int l = strlen(Input);
[LO8]  228:                    for (i=0 ; i<l ; i+=OutTabP
       os)
[V30]  229:                    {
[5Z4]  230:                        int k=OutTabPos;
[WW8]  231:                        strncpy(Output, Input+i
       , k);
[XG4]  232:                        Output[k] = '\0';
[XA8]  233:                        while (k && Output[--k]
        == SPACE)
[AN9]  234:                            Output[k] = TAB, Ou
       tput[k+1]='\0';
[5X8]  235:                        if (Output[OutTabPos-1]
        == TAB)
[1V7]  236:                            Output[OutTabPos-1]
        = SPACE;
[TM9]  237:                                          /*
       Ausgabe mit Tabulatoren                  */
[PK5]  238:                        printf("%s", Output);
[X30]  239:                    }
[GB3]  240:                    printf("\n");
[X30]  241:                }
[4RE]  242:                i=0, ++Line;              /*
       Zeichenzaehler zuruecksetzen             */
[X30]  243:            }
[X30]  244:        }
[LGB]  245: }                                        /*
       end of work()                            */
[Z8C]  246: /*-------------------------------------
       ------------------------------------*/
[000]  247:
[045]  248:                                          /*
       <f> 15                                   */
[000]  249:
[Z8C]  250: /*-------------------------------------
       ------------------------------------*/
[9HC]  251: /* Funktion     : Fehler()
                                                 */
[Z8C]  252: /*-------------------------------------
       ------------------------------------*/
[E4K]  253: /* Beschreibung : <c> Fehlermeldungen ausge
       ben                                       */
[9V8]  254: /*
                                                 */
[Z8C]  255: /*-------------------------------------
       ------------------------------------*/
[000]  256:
[UUE]  257: void Fehler(FehlerNr, FehlerText)
                                                 /*
[LOC]  258: ================================
                                                 */
[RP3]  259: int FehlerNr;
[OV4]  260: char *FehlerText;
[V30]  261: {
[L45]  262:     switch (FehlerNr)
[V30]  263:     {
[ZE3]  264:         case OPT_ERR :
[5X5]  265:             fprintf (stderr,
[ZEA]  266:                      "rpltab : fehlerhafte
       Option : %s\n",
[GD3]  267:                      FehlerText);
[KV1]  268:             break;
[000]  269:
[2C3]  270:         case CMD_ERR :
[5Y5]  271:             fprintf (stderr,
[X7E]  272:                      "rpltab : Fehler in de
       r Kommandozeile : %s\n",
[GD3]  273:                      FehlerText);
[KV1]  274:             break;
[000]  275:
[0C3]  276:         case LEN_ERR :
[5Y5]  277:             fprintf (stderr,
[5EA]  278:                      "rpltab : zu lang Text
       zeile %d\n",
[BH1]  279:                      Line);
[KV1]  280:             break;
[000]  281:
[XS3]  282:         default      :
[5X5]  283:             fprintf (stderr,
[PNC]  284:                      "rpltab : unbekannte F
       ehlermeldung : %d\n",
[YU2]  285:                      FehlerNr);
[5X5]  286:             fprintf (stderr,
[EW6]  287:                      "Programmabbruch!\n");
[8Y1]  288:             exit(-1);
[000]  289:
[LDC]  290:     }                                    /*
       Ende switch(FehlerNr)                    */
[HRB]  291: }                                       /*
       end of Fehler()                          */
[Z8C]  292: /*-------------------------------------
       ------------------------------------*/
[000]  293:
[045]  294:                                          /*
       <f> 15                                   */
[000]  295:
[Z8C]  296: /*-------------------------------------
       ------------------------------------*/
[YGC]  297: /* Funktion     : Hilfe()
                                                 */
[Z8C]  298: /*-------------------------------------
       ------------------------------------*/
[YEM]  299: /* Beschreibung : <c> Programmhilfe am Bild
       schirm ausgeben                          */
[9V8]  300: /*
                                                 */
[Z8C]  301: /*-------------------------------------
       ------------------------------------*/
[000]  302:
[3KA]  303: void Hilfe()
                                                 /*
[MW9]  304: ============
                                                 */
[V30]  305: {
[1E6]  306: # include "rpltabh.i"
[YAB]  307: }                                       /*
       end of Help()                            */
[Z8C]  308: /*-------------------------------------
       ------------------------------------*/
[000]  309:
[000]  310:
[NFB]  311: /*************************************
       ************************************/
[X6B]  312: /*                          E n d e   r p l
       t a b . c                                 */
[NFB]  313: /*************************************
       ************************************/
```

Listing 1. Der Quellcode zu »rpltab.c«

```
[43M]    0: printf("Aufruf : >rpltab [-?|-H] [-S] [-In]
            [-On]  [I/O-Umleitung]\n");
[GB3]    1: printf("\n");
[WPV]    2: printf("Beschreibung :  'rpltab' liest zeil
            enweise Text vom standart Eingabegeraet\n");
[83T]    3: printf("                und ersetzt dabei a
            lle Tabulatoren durch Leerzeichen mit der\n"
            );
[2AS]    4: printf("                Schrittweite 8. Vor
            der Ausgabe wiederum werden alle Leer-\n");
[25S]    5: printf("                zeichen durch Tabul
            atoren mit der Schrittweite 8 ersetzt.\n");
[GB3]    6: printf("\n");
[GB3]    7: printf("\n");
[Y8K]    8: printf("Optionen     : -?,-H  Ausgabe dies
            er Hilfsinformation.\n");
[GB3]    9: printf("\n");
[XFN]   10: printf("                -S     Bei der Ausg
            abe werden keine Tabulatoren gesetzt,\n");
[71K]   11: printf("                       es werden nu
            r Leerzeichen ausgegeben.\n");
[GB3]   12: printf("\n");
[VKN]   13: printf("                -In    Beim Einlese
            n des Textes werden Tabulatoren mit\n");
[HBP]   14: printf("                       der Schrittw
            eite n beruecksichtigt, z.B. rpltab -010.\n"
            );
[GB3]   15: printf("\n");
[GKN]   16: printf("                -On    Bei der Ausg
            abe des Textes werden Tabulatoren mit\n");
[R8L]   17: printf("                       der Schrittw
            eite n gesetzt, z.B. rpltab -04.\n");
```

Listing 2. Der Quellcode der Include-Datei »rpltabh.i« enthält den Hilfstext

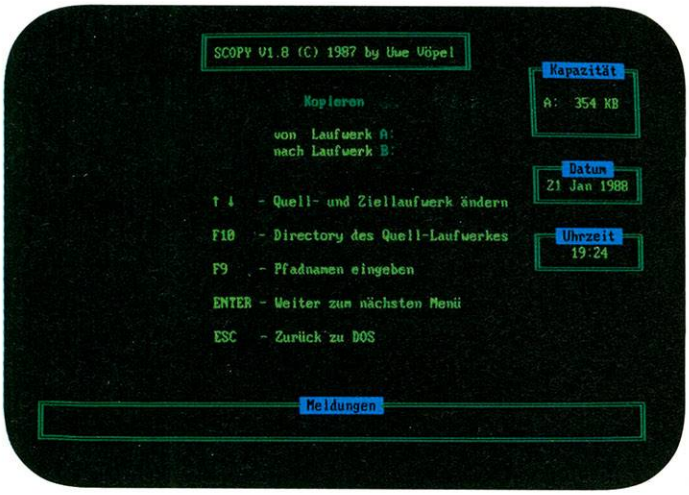


Bild 1. Per Menü werden Sie durch SCopy geführt

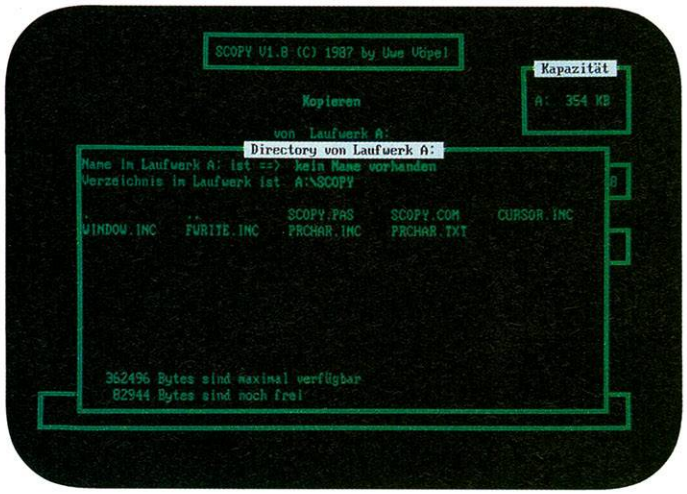


Bild 2. Mit [Alt-F10] überprüfen Sie die Zieldiskette

SCopy, das professionelle Kopierprogramm

Wer kennt sie nicht, die umständlichen und zeitaufwendigen Kopieranweisungen, um ein paar Dateien von Diskette zu Diskette oder auf die Festplatte zu kopieren. SCopy nimmt Ihnen diese lästige Arbeit ab.

SCopy ist in Turbo Pascal geschrieben (Listing 1). Es dient dem komfortablen Kopieren von Dateien oder Programmen auf Disketten und Festplatte. SCopy wird mit oder ohne Parameter gestartet. Die Syntax lautet:

```
scopy <Quellaufwerk> <Ziellaufwerk>
```

Lassen Sie beide Parameter weg, so setzt das Programm in beiden Fällen automatisch das aktuelle Laufwerk ein. Natürlich müssen Sie in solch einem Fall die Laufwerksangabe ändern, da das Kopieren auf einer Diskette sinnlos und deshalb auch nicht gestattet ist. Der Befehl

```
scopy A: B:
```

setzt A als Quellaufwerk und B als Ziellaufwerk ein.
Das Programm bedarf keiner großen Erklärung, da es per Menü gesteuert wird (Bild 1). Zum besseren Verständnis werden im folgenden trotzdem die wichtigsten Befehle erklärt.
Unter dem Fenster mit dem Programmnamen und der Copyrightmeldung steht das Hauptmenü. Hier können Sie mit den Cursor-Steuertasten [Cursor-aufwärts] und [Cursor-abwärts] das Quell- und Ziellaufwerk verändern. Zur Auswahl stehen die Laufwerksbezeichnungen A bis maximal F, sofern Sie so viele Laufwerke angemeldet haben. Die Buchstaben sind nämlich auf die Zahl der Laufwerke beschränkt, über die Ihr Computer auch tatsächlich verfügt.
Durch Drücken der F9-Taste gelangen Sie in ein Eingabefenster, wo der Pfadname für Quell- oder Ziellaufwerk eingegeben wird. Benutzern von MS-DOS-Versionen, die älter sind als 3.0, müssen bei der Eingabe des Pfadnamens den »Backslash« (ASCII-Wert 92) benutzen. Versionen ab 3.0 akzeptieren sowohl den Schrägstrich als auch den »Backslash«.
Die F10-Taste zeigt das Inhaltsverzeichnis des Quellaufwerks an, [Alt-F10] hingegen, das Directory des »Ziellaufwerks« (Bild 2). Nach Betätigen der Enter-Taste wird im linken Teil des Bildschirms ein Auswahlfenster mit dem Directory des Quellaufwerks ausgegeben (Bild 3). Der Cursor steht auf der ersten Datei im Auswahlfenster. Gleichzeitig wird ein neues Untermenü ausgegeben:

Cursor-aufwärts – Cursor eine Zeile höher
Cursor-abwärts – Cursor eine Zeile tiefer
PgUp/PgDn – eine Seite zurück-, beziehungsweise vorblättern
F10 – weitere Optionen
Leertaste – markiert die Datei für den Kopiervorgang
Enter – markierte Dateien werden kopiert
Esc – zurück zum ersten Menü
Home – Cursor an den Anfang im Auswahlfenster stellen
End – Cursor an das Ende im Auswahlfenster stellen

Die beiden Funktionen »Home« und »End« sind in dem Menü nicht explizit aufgeführt, was aber nicht heißt, daß Sie sie nicht benutzen dürfen. Zusätzlich läßt sich durch Drücken der Alt-Taste gemeinsam mit einem Buchstaben der Cursor direkt auf der ersten Datei mit diesem Anfangsbuchstaben positionieren.

Ein weiters Menü aktiviert die Taste [F10] (Bild 4). Es bietet folgende Optionen:

Del – Löschen der markierten Daten nach vorheriger Abfrage
M – Markieren aller Dateien
U – Markierung aller Dateien wird aufgehoben
N – speziellen Dateinamen eingeben
F5 – Ausgabe der Speicherkapazität des Quellaufwerks
Alt-F5 – Ausgabe der Speicherkapazität des Ziellaufwerks

Beim Menü-Punkt »N« werden spezielle Dateien ausgewählt. Erlaubte Kriterien sind Eingaben wie »*.pas« oder »b*.com«. Das Programm markiert dann alle entsprechenden Dateien. Für die Ein- und Ausgabe der Menü-Punkte »N«, »F5« und »Alt-F5« steht ein eigenes Bildschirmfenster zur Verfügung. Alle Fenster und Menüs werden mit der Esc-Taste wieder verlassen.

Wenn Sie im zweiten Menü die Enter-Taste drücken, erscheinen im Auswahlfenster die zu kopierenden Dateien. Stellt SCopy während des Kopiervorgangs fest, daß die zu kopierende Datei auf der Zieldiskette bereits vorhanden ist, erwartet es eine Bestätigung, ob die Datei überschrieben werden darf. Nach dem Kopieren erfolgt die Rückkehr ins Hauptmenü, das Sie wiederum mit [Esc] beenden.

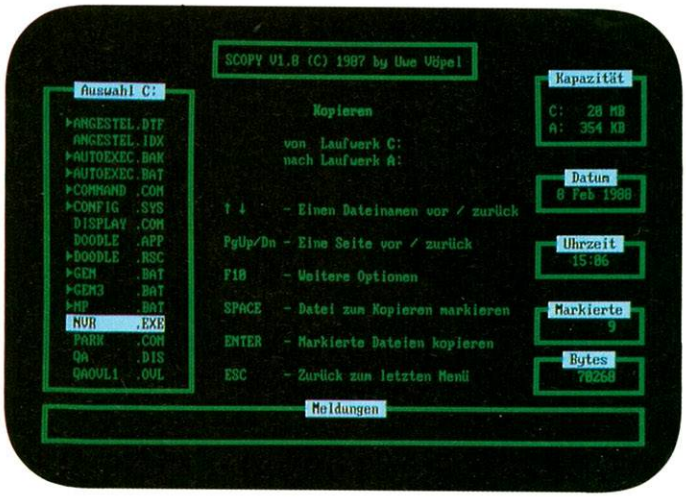


Bild 3. Auf komfortable Weise werden die Dateien selektiert

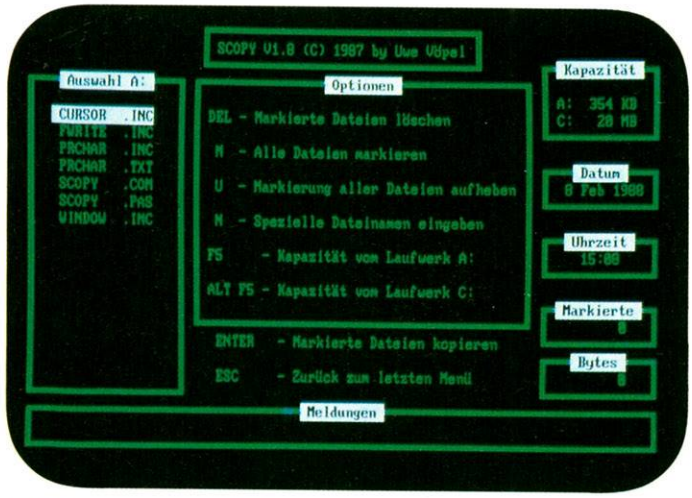


Bild 4. Das Optionen-Menü von SCopy

Sie sollten das Programm grundsätzlich über des Hauptmenü verlassen, damit der angelegte reservierte Speicher (Heap) wieder freigegeben wird. Als Sicherung gegen eine vorzeitige Unterbrechung integrieren Sie am besten die Compilerdirektive »C – ae- (Zeile 9 in Listing 1) in das Programm, denn ein Abbruch während des Kopiervorgangs zerstört mit Sicherheit einige Dateien auf Diskette oder Festplatte.

Die Include-Dateien (Listing 2 bis 5 – sie müssen vor dem Compilieren mit auf der Diskette stehen) lassen sich auch in andere Programme integrieren. Die Routinen für die Fenster-Verwaltung finden Sie in der Include-Datei »window.inc« (Listing 3). Für die Fenster-Routine werden zwei Konstantenvereinbarungen benutzt:

BildsBuffer – Reservieren von Speicherplatz zum Speichern des Bildschirmhintergrunds

MaxFenster – Zahl der maximal gleichzeitig geöffneten Fenster

Zum Benutzen der Fenster werden allerdings nur die Routinen »FensterAuf« und »FensterZu« benötigt. »FensterZu« schließt jeweils das zuletzt geöffnete Fenster, wodurch sich eine Parameterübergabe für diese Routine erübrigt.

Die Prozedur »FensterAuf« rufen Sie mit folgenden Parametern auf:

```
FensterAuf( 'Überschrift',x1,y1,x2,y2 );
```

Die Parameter »x1« und »x2« kennzeichnen die Anfangs- und Endpunkte der Bildschirmspalte. Die Start- und Endzeile des Fensters wird durch die Parameter »y1« und »y2« übergeben. Die Verwendung von Fenster-Routinen erfordert die Include-Datei »fwrite.inc« für eine schnellere Bildschirmausgabe.

Die Eingabe-Routine (Input) wird durch eine Funktion, die beliebig erweitert werden kann, realisiert. Die Syntax dieser Funktion mit ihren Parametern lautet:

```
Variable:= Input( x, y, Laenge );
```

Die Eingabe erfolgt ab der Bildposition, die durch die Parameter »x« und »y« definiert wird. Der Parameter »Laenge« gibt die maximale Zahl der einzugebenden Zeichen an. Die normalen Editierfunktionen wie [Home], [End], [Ins], [Del], [Cursor-links] und [Cursor-rechts] werden in dieser Eingabe-Routine ganz normal benutzt. [Esc] bricht die Eingabe ab.

Die Input-Funktion läßt sich mit der Routine zur schnelleren Bildschirmausgabe (Listing 4, »fwrite.inc«) nicht verbessern, da sich die Fwrite-Prozedur keiner relativen Koordinaten (in Bezug auf ein geöffnetes Fenster) bedient.

Für den Fall, daß Sie zusätzliche Informationen zu einem Menü benötigen, bietet das Programm eine Hilfsfunktion, die mit der F1-Taste aufgerufen wird. Zu jedem Menü existiert ein eigener kleiner Hilfsbildschirm, der Hinweise zum jeweiligen Menü enthält.

(Uwe Vöpel/sp)

```
[7YA]    0: (*****************************************
           *********************************)
[UDH]    1: (*           SUPERCOPY     Version 1.8   (C)
         Dezember   1987 by Uwe Vöpel     *)
[UZE]    2: (*                          Datum der letzten
         Änderung:  26.12.1987             *)
[KND]    3: (*                          Betriebssystem:
                    PC / MS-DOS           *)
[S3D]    4: (*                          Compiler:
                    Turbo-Pascal 3.01     *)
[7YA]    5: (*****************************************
           *********************************)
[000]    6:
[N16]    7: program Super_Copy;
[000]    8:
[G71]    9: (*$C-*)
[000]   10:
[VM1]   11: const
[WL4]   12:         BildsBuffer = 2000;
[064]   13:         MaxFenster  = 3;
[EWB]   14:         CopyRight   = 'SCOPY V1.8 (C) 1987 b
         y Uwe Vöpel';
[M34]   15:         MaxZahl     = 512;
[NH3]   16:         MaxRec      = 511;
```

```
[EF3]   17:         FFFF        = 65536.0;
[000]   18:
[A21]   19: type
[4U5]   20:         Index       = 0..MaxZahl;
[C54]   21:         Cursor      = 1..16;
[645]   22:         KB_MB       = string(.7.);
[F55]   23:         Time        = string(.5.);
[U85]   24:         Date        = string(.11.);
[L65]   25:         Str12       = string(.12.);
[VC6]   26:         FileName    = string(.14.);
[P85]   27:         Pfad        = string(.64.);
[P85]   28:         Str80       = string(.80.);
[7N6]   29:         Meldung     = string(.76.);
[KM5]   30:         DirEintrag  = record
[C05]   31:                         FName     : FileName;
[R65]   32:                         FLaenge   : real;
[XX5]   33:                         Markiert  : boolean;
[RY3]   34:                         DName     : Str12
[TU0]   35:                       end;
[B85]   36:         BildRec     = record
[805]   37:                         BName     : FileName;
[Y35]   38:                         SubDir    : boolean
[TU0]   39:                       end;
[5H5]   40:         Register    = record
```

```
[S25]   41:                       case integer of
[T4B]   42:                         1:(AX,BX,CX,DX,BP,SI
        ,DI,DS,ES,Flags : integer);
[LD7]   43:                         2:(AL,AH,BL,BH,CL,CH
        ,DL,DH : byte);
[TUO]   44:                       end;
[9XB]   45:       Buffer      = array(.-MaxInt..MaxInt
        .) of byte;
[000]   46:
[EMO]   47: var
[33C]   48:       Directory   : array(.Index.) of DirE
          intrag;
[H3B]   49:       BildName    : array(.Index.) of Bild
          Rec;
[5K2]   50:       AnZahl, z,
[SO4]   51:       BZahl       : Index;
[1A5]   52:       REGS        : Register;
[M36]   53:       BufferPtr   : ^Buffer;
[935]   54:       Nr          : ^integer;
[UF3]   55:       QF, ZF      : file;
[BV1]   56:       Bytes,
[695]   57:       Mark        : string(.10.);
[761]   58:       Name,
[XS3]   59:       TName       : Str12;
[KR2]   60:       RestRec,
[KA2]   61:       AnzRec,
[7D4]   62:       MK          : integer;
[HE2]   63:       Fehler,
[E84]   64:       ADrive      : byte;
[SE3]   65:       TAnz        : 0..550;
[OW3]   66:       CNr         : Cursor;
[9P1]   67:       QFile,
[KP1]   68:       ZFile,
[3N1]   69:       QPfad,
[GL3]   70:       ZPfad       : Pfad;
[3R2]   71:       QFrei, QMax,
[CR2]   72:       ZFrei, ZMax,
[TO3]   73:       KB          : real;
[OS3]   74:       QKB, ZKB    : KB_MB;
[HR2]   75:       Eingabe,
[ZA2]   76:       JaNein,
[YRO]   77:       QLw,
[634]   78:       ZLw, Ein, C : char;
[W95]   79:       Ausg        : string(.20.);
[NN5]   80:       Titel       : string(.30.);
[NB4]   81:       MSG         : Meldung;
[Z92]   82:       LwFlag,
[FV4]   83:       Flag        : boolean;
[000]   84:
[BNA]   85: (*$I FWRITE.INC *)    (*********************
        ************************)
[VXH]   86: (*$I PRCHAR.INC *)    (*   Include-Dateien
        ohne die das Programm  *)
[88E]   87: (*$I CURSOR.INC *)    (*              nicht
        funktioniert !         *)
[PPA]   88: (*$I WINDOW.INC *)    (*********************
        ************************)
[000]   89:
[7YA]   90: (*************************************
        *********************************)
[M39]   91: (*                          H E L P  - 1 -
                                                  *)
[7YA]   92: (*************************************
        *********************************)
[000]   93:
[2D5]   94: procedure  Help1;
[000]   95:
[EMO]   96: var
[EB2]   97:         Ein : char;
[000]   98:
[BH1]   99: begin
[L5A]  100:   FensterAuf(' Meine Adresse ',22,5,57,22);
[7UE]  101:   fwrite(24, 6,2,'Bei programmtechnischen F
        ragen :');
[097]  102:   fwrite(27, 8,10,'Uwe Vöpel');
[YAB]  103:   fwrite(27, 9,10,'Karolinenstraße 41');
[R38]  104:   fwrite(27,10,10,'4600 Dortmund 76');
[4F9]  105:   fwrite(27,11,10,'Telefon: 0231 / 61 47 47
        ');
[SXD]  106:   fwrite(21,13,10,'╠'); PrChar(22,13,10,'='
        ,36); fwrite(58,13,10,'╣');
[D8A]  107:   fwrite(31,13,112,' Hinweis zum Menü ');
[ARE]  108:   fwrite(23,15,2,'ALT F10 ==> Anzeige des D
        irectory''s');
[GL9]  109:   fwrite(35,16,2,'vom Ziellaufwerk');
[5ZD]  110:   fwrite(21,18,10,'╠'); PrChar(22,18,10,'='
        ,36); fwrite(58,18,10,'╣');
[LC7]  111:   fwrite(35,18,112,' Achtung ! ');
[MHB]  112:   fwrite(27,20,2,'Quell- und Ziellaufwerk')
        ;
[5ZC]  113:   fwrite(25,21,2,'müssen unterschiedlich se
        in !');
[GR7]  114:   CursorAus; read(Kbd,Ein);
[H63]  115:   FensterZu;
[TUO]  116: end;
```

```
[000]  117:
[7YA]  118: (*************************************
       *********************************)
[U69]  119: (*                                H E L P  - 2
                                                  *)
[7YA]  120: (*************************************
       *********************************)
[000]  121:
[2D5]  122: procedure  Help2;
[000]  123:
[EMO]  124: var
[EB2]  125:         Ein : char;
[000]  126:
[BH1]  127: begin
[2R7]  128:   FensterAuf(' Hilfe ',24,8,59,18);
[F5D]  129:   fwrite(27,9,10,'Weitere Tastenbelegungen
       :');
[NYC]  130:   fwrite(27,11,2,'HOME  ==> Cursor an den A
       nfang');
[S5C]  131:   fwrite(27,12,2,'END   ==> Cursor an das E
       nde');
[GY9]  132:   fwrite(27,14,2,'ALT und Buchstabe:');
[9VB]  133:   fwrite(31,15,2,'Cursor wird auf die Datei
       ');
[9PC]  134:   fwrite(31,16,2,'mit den Anfangsbuchstaben
       ');
[KM8]  135:   fwrite(31,17,2,'positioniert.');
[GR7]  136:   CursorAus; read(Kbd,Ein);
[V23]  137:   FensterZu
[TUO]  138: end;
[000]  139:
[7YA]  140: (*************************************
       *********************************)
[W69]  141: (*                                H E L P  - 3
                                                  *)
[7YA]  142: (*************************************
       *********************************)
[000]  143:
[2D5]  144: procedure  Help3;
[000]  145:
[EMO]  146: var
[EB2]  147:         Ein : char;
[000]  148:
[BH1]  149: begin
[8VA]  150:   FensterAuf(' Hilfe (Optionen) ',24,6,59,1
       8);
[2MD]  151:   fwrite(25,7,10,'Bedeutung der Funktionen
       M, U, N:');
[POD]  152:   fwrite(26, 9,2,'M: Es werden alle Dateien
       zum');
[K7A]  153:   fwrite(29,10,2,'Kopieren markiert.');
[8TC]  154:   fwrite(26,11,2,'U: Die Markierung der Dat
       eien');
[HG9]  155:   fwrite(29,12,2,'wird aufgehoben.');
[1AC]  156:   fwrite(26,13,2,'N: Es werden nur bestimmt
       e');
[S1A]  157:   fwrite(29,14,2,'Dateien markiert.');
[FVB]  158:   fwrite(29,15,2,'(z.B.: *.EXE -> es werden
       alle');
[9UD]  159:   fwrite(29,16,2,'Dateien mit der Extension
       EXE');
[F2B]  160:   fwrite(29,17,2,'zum Kopieren markiert)');
[GR7]  161:   CursorAus; read(Kbd,Ein);
[V23]  162:   FensterZu
[TUO]  163: end;
[000]  164:
[7YA]  165: (*************************************
       *********************************)
[VLB]  166: (*                        Block - Cursor setzen
                                                  *)
[7YA]  167: (*************************************
       *********************************)
[000]  168:
[OZ7]  169: procedure  CursorBlock;
[000]  170:
[BH1]  171: begin
[F13]  172:   with REGS do
[BH1]  173:   begin
[A61]  174:     AH:=1;
[7R3]  175:     if Test then
[BH1]  176:     begin
[K37]  177:       CH:=1;              (*              für
                                        *)
[C2D]  178:       CL:=13              (*  Monochrom-Adapt
       er (Hercules)  *)
[DLO]  179:     end
[A11]  180:     else
[BH1]  181:     begin
[D37]  182:       CH:=0;              (*              für
                                        *)
[67B]  183:       CL:=7               (*     Color-Graphi
       k-Adapter      *)
[TUO]  184:     end;
[GV2]  185:     INTR($10,REGS)
[DLO]  186:   end
```

```
[TUO]  187: end;
[000]  188:
[7YA]  189: (*****************************************
       ********************************)
[W2C]  190: (*                        Normal - Cursor setze
       n                                *)
[7YA]  191: (*****************************************
       ********************************)
[000]  192:
[UA8]  193: procedure  CursorNormal;
[000]  194:
[BH1]  195: begin
[F13]  196:   with REGS do
[BH1]  197:   begin
[A61]  198:     AH:=1;
[7R3]  199:     if Test then
[BH1]  200:     begin
[T77]  201:       CH:=12;              (*              für
                                        *)
[C2D]  202:       CL:=13               (*  Monochrom-Adapt
       er (Hercules)  *)
[DLO]  203:     end
[A11]  204:     else
[BH1]  205:     begin
[B47]  206:       CH:=6;               (*              für
                                        *)
[67B]  207:       CL:=7                (*     Color-Graphi
       k-Adapter      *)
[TUO]  208:     end;
[GV2]  209:     INTR($10,REGS)
[DLO]  210:   end
[TUO]  211: end;
[000]  212:
[7YA]  213: (*****************************************
       ********************************)
[XUF]  214: (*                        Testen der Laufwerke
       (nur A: oder B:)                 *)
[7YA]  215: (*****************************************
       ********************************)
[000]  216:
[PR9]  217: function  TestLW( LW : char ) : byte;
[000]  218:
[BH1]  219: begin
[K36]  220:   if (LW='A') or (LW='B') then
[BH1]  221:   begin
[F13]  222:     with REGS do
[BH1]  223:     begin
[RH8]  224:       AH:=$0E; DL:=ord(LW)-65; MsDos(REGS);
[YC8]  225:       AH:=4; DL:=ord(LW)-65; DH:=0; CH:=0;
       CL:=1;
[UX6]  226:       INTR(19,REGS); AH:=1; INTR(19,REGS);
[CL3]  227:       if AH > 0 then
[BH1]  228:       begin
[7U6]  229:         if AH > 63 then TestLW:=128
[A11]  230:         else
[BB2]  231:           case AH of
[YC3]  232:             2: TestLW:=2;
[3D3]  233:             3: TestLW:=3;
[FP3]  234:             else TestLW:=0
[DLO]  235:           end
[DLO]  236:       end
[C34]  237:       else TestLW:=0;
[TUO]  238:     end;
[DLO]  239:   end
[C34]  240:   else TestLW:=0;
```

```
[TUO]  241: end;
[000]  242:
[7YA]  243: (*****************************************
       ********************************)
[7KM]  244: (*                  Anzeige von allgemeinen Mel
       dungen und Fehlermeldungen       *)
[7YA]  245: (*****************************************
       ********************************)
[000]  246:
[HFA]  247: procedure  Message( MSG : Meldung );
[000]  248:
[A21]  249: type
[6G7]  250:       Str78 = array(.1..78.) of char;
[000]  251:
[EMO]  252: var
[A73]  253:       Leer  : Str78;
[000]  254:
[BH1]  255: begin
[XS5]  256:   fillchar(Leer,78,' ');
[NK3]  257:   if MSG <> ' ' then
[BH1]  258:   begin
[20C]  259:     CursorAus; sound(2000); delay(250); nos
       ound
[TUO]  260:   end;
[7N4]  261:   fwrite(1,23,2,Leer);
[K8C]  262:   fwrite(78 div 2-length(MSG) div 2+1,23,13
       8,MSG);
[TUO]  263: end;
[000]  264:
[7YA]  265: (*****************************************
       ********************************)
[68C]  266: (*                        Eingabe - Routine
                                        *)
[7YA]  267: (*****************************************
       ********************************)
[000]  268:
[VYC]  269: function  Input( x,y,Len : byte ) : Str80;
[000]  270:
[EMO]  271: var
[943]  272:       Z         : char;
[EO4]  273:       Temp      : Str80;
[V44]  274:       x2, y2, i : byte;
[1U5]  275:       InsFlag   : boolean;
[000]  276:
[BH1]  277: begin
[BGB]  278:   x2:=pred(x); y2:=y; x:=1; InsFlag:=false;
[AU5]  279:   fillchar(Temp,81,' ');
[E45]  280:   gotoxy(succ(x2),y2);
[X52]  281:   repeat
[EV2]  282:     read(Kbd,Z);
[HM8]  283:     if (Z=#27) and keypressed then
[BH1]  284:     begin
[EV2]  285:       read(Kbd,Z);
[G64]  286:       case ord(Z) of
[ZH7]  287:        75: begin    (* Cursor links *)
[K13]  288:              x:=pred(x);
[TA4]  289:              if x < 1  then x:=1;
[P44]  290:              gotoxy(x2+x,y)
[TUO]  291:            end;
[EW7]  292:        77: begin    (* Cursor rechts *)
[U13]  293:              x:=succ(x);
[ZH6]  294:              if x > Len then x:=Len;
[P44]  295:              gotoxy(x2+x,y)
[TUO]  296:            end;
[OP4]  297:        71: begin    (* HOME *)
```


Lieben Sie Parties? – Wir planen bereits Ihre Einstandsfeier

Wenn Sie Interesse haben, in einem jungen unkoventionellen Team in der Computerbranche mitzuarbeiten, dann sind Sie bei uns richtig. Wir suchen Fachredakteure mit Berufserfahrung für die Aufgabengebiete

* Software-Anwendungen
* Programmiersprachen
* Betriebssysteme

Wir bieten Ihnen Freiräume für die Umsetzung eigener Ideen. Eine leistungsorientierte Vergütung, flexible Arbeitszeit und eine zusätzliche Altersversorgung sind bei uns selbstverständlich.
Von Ihnen erwarten wir Einsatzbereitschaft und Eigeninitiative. Bitte richten Sie Ihre Bewerbung mit den üblichen Unterlagen und einigen Arbeitsproben an

Redaktionsbüro Everts & Hagedorn
Redaktion DOS International
Elfenstr. 40
8000 München 83

```
[EWO]  298:                x:=1;
[P44]  299:                gotoxy(x2+x,y)
[TUO]  300:              end;
[994]  301:         79: begin     (* END *)
[X72]  302:                x:=Len;
[P44]  303:                gotoxy(x2+x,y)
[TUO]  304:              end;
[DA4]  305:         82: begin     (* INS *)
[225]  306:                if InsFlag then
[BH1]  307:                begin
[E44]  308:                  CursorNormal;
[2U4]  309:                  InsFlag:=false
[DLO]  310:                end
[A11]  311:                else
[BH1]  312:                begin
[PK3]  313:                  CursorBlock;
[H84]  314:                  InsFlag:=true
[TUO]  315:                end;
[TUO]  316:              end;
[394]  317:         83: begin     (* DEL *)
[HF4]  318:                delete(Temp,x,1);
[S46]  319:                insert('  ',Temp,Len);
[NT9]  320:                for i:=x to Len do write(Temp(
       .i.));
[P44]  321:                gotoxy(x2+x,y)
[TUO]  322:              end;
[TUO]  323:         end;
[DLO]  324:       end
[A11]  325:       else
[V6C]  326:         if Z >= #32 then if Z <= #128 then if
        x <= Len then
[BH1]  327:         begin
[225]  328:           if InsFlag then
[BH1]  329:           begin
[E55]  330:             insert(Z,Temp,x);
[NT9]  331:             for i:=x to Len do write(Temp(.i.
       ));
[U13]  332:             x:=succ(x);
[ZH6]  333:             if x > Len then x:=Len;
[P44]  334:             gotoxy(x2+x,y)
[DLO]  335:           end
[A11]  336:           else
[BH1]  337:           begin
[C93]  338:             Temp(.x.):=Z;
[U13]  339:             x:=succ(x);
[OG2]  340:             write(Z);
[ZH6]  341:             if x > Len then x:=Len;
[P44]  342:             gotoxy(x2+x,y)
[TUO]  343:           end;
[TUO]  344:         end;
[S23]  345:         if Z=#27 then
[BH1]  346:         begin
[T83]  347:           Temp(.1.):=#27;
[S11]  348:           Z:=#13
[DLO]  349:         end
[A11]  350:         else
[S23]  351:           if Z=#25 then
[BH1]  352:           begin
[EWO]  353:             x:=1;
[857]  354:             gotoxy(x2+x,y); clreol;
[8F5]  355:             fillchar(Temp,81,' ')
[DLO]  356:           end
[A11]  357:           else
[LN2]  358:             if Z=#8 then
[BH1]  359:             begin
[7F4]  360:               if pred(x) > 0 then
[BH1]  361:               begin
[9L6]  362:                 delete(Temp,pred(x),1);
[S46]  363:                 insert('  ',Temp,Len);
[3V5]  364:                 gotoxy(x2+pred(x),y);
[VKA]  365:                 for i:=pred(x) to Len do writ
       e(Temp(.i.));
[K13]  366:                 x:=pred(x);
[P44]  367:                 gotoxy(x2+x,y)
[TUO]  368:               end;
[TUO]  369:             end;
[DX2]  370:     until Z=#13;
[E44]  371:     CursorNormal;
[313]  372:     Input:=Temp
[TUO]  373: end;
[000]  374:
[7YA]  375: (*****************************************
       *********************************)
[55E]  376: (*                        Datum vom System erfr
       agen                          *)
[7YA]  377: (*****************************************
       *********************************)
[000]  378:
[ZW7]  379: function  Datum : Date;
[000]  380:
[EMO]  381: var
[PR4]  382:       Jahr  : string(.4.);
[645]  383:       Monat : string(.3.);
[LD4]  384:       Tag   : string(.2.);
[000]  385:
```

```
[BH1]  386: begin
[F13]  387:   with REGS do
[BH1]  388:   begin
[9P4]  389:     AH:=$2A; MsDos(REGS);
[VK7]  390:     str(CX:4,Jahr); str(DL:2,Tag);
[XB2]  391:     case DH of
[41E]  392:       1 : Monat:='Jan';   2 : Monat:='Feb';
       3 : Monat:='Mär';
[HOE]  393:       4 : Monat:='Apr';   5 : Monat:='Mai';
       6 : Monat:='Jun';
[S1E]  394:       7 : Monat:='Jul';   8 : Monat:='Aug';
       9 : Monat:='Sep';
[F7E]  395:      10: Monat:='Okt';   11: Monat:='Nov';
      12: Monat:='Dez';
[TUO]  396:     end;
[TUO]  397:   end;
[WR6]  398:   Datum:=Tag+' '+Monat+' '+Jahr
[TUO]  399: end;
[000]  400:
[7YA]  401: (*****************************************
       *********************************)
[K1F]  402: (*                        Uhrzeit vom System erf
       ragen                         *)
[7YA]  403: (*****************************************
       *********************************)
[000]  404:
[LZ7]  405: function  Uhrzeit : Time;
[000]  406:
[EMO]  407: var
[G42]  408:       Std, Min,
[ES5]  409:       Sek, Sek100 : string(.2.);
[000]  410:
[BH1]  411: begin
[F13]  412:   with REGS do
[BH1]  413:   begin
[RP4]  414:     AH:=$2C; MsDos(REGS);
[ZW5]  415:     str(CH:2,Std); str(CL,Min);
[RW9]  416:     if length(Min) = 1 then Min:='0'+Min;
[478]  417:     if CH=$00 then fwrite(67,9,2,Datum)
[TUO]  418:   end;
[306]  419:   Uhrzeit:=Std+':'+Min;
[TUO]  420: end;
[000]  421:
[7YA]  422: (*****************************************
       *********************************)
[YWB]  423: (*                          Bildschirm - Aufbau
                                       *)
[7YA]  424: (*****************************************
       *********************************)
[000]  425:
[4K5]  426: procedure  Screen;
[000]  427:
[EMO]  428: var
[P82]  429:         x  : byte;
[000]  430:
[BH1]  431: begin
[OTF]  432:   fwrite(22,0,10,'╔'); PrChar(23,0,10,'=',3
       4);     fwrite(57,0,10,'╗');
[T4H]  433:   fwrite(22,1,10,'║'); fwrite(24,1,2,CopyRi
       ght);   fwrite(57,1,10,'║');
[FTF]  434:   fwrite(22,2,10,'╙'); PrChar(23,2,10,'=',3
       4);     fwrite(57,2,10,'╜');
[UY6]  435:   fwrite(36,4,10,'Kopieren');
[FZ7]  436:   fwrite(32,6,2,'von  Laufwerk  :');
[658]  437:   fwrite(32,7,2,'nach Laufwerk  :');
[7W9]  438:   fwrite(46,6,10,QLw); fwrite(46,7,10,ZLw);
[X23]  439: (*  1. Rahmen *)
[NCF]  440:   fwrite(65,2,10,'╔'); PrChar(66,2,10,'=',1
       3); fwrite(79,2,10,'╗');
[W48]  441:   fwrite(67,2,112,' Kapazität ');
[LY3]  442:   for x:=3 to 5 do
[BH1]  443:   begin
[5L4]  444:     fwrite(65,x,10,'│');
[XB4]  445:     fwrite(79,x,10,'│')
[TUO]  446:   end;
[6DF]  447:   fwrite(65,6,10,'╙'); PrChar(66,6,10,'=',1
       3); fwrite(79,6,10,'╜');
[LT5]  448:   fwrite(67,4,2,QLw+': '+QKB);
[233]  449: (*  2. Rahmen *)
[XMF]  450:   fwrite(0,22,10,'╔'); PrChar(1,22,10,'=',7
       8); fwrite(79,22,10,'╗');
[3V7]  451:   fwrite(35,22,112,' Meldungen ');
[MZ9]  452:   fwrite(0,23,10,'│'); fwrite(79,23,10,'│')
       ;
[BNF]  453:   fwrite(0,24,10,'╙'); PrChar(1,24,10,'=',7
       8); fwrite(79,24,10,'╜');
[4N2]  454: (*  Datum  *)
[7DF]  455:   fwrite(65,8,10,'╔'); PrChar(66,8,10,'=',1
       3); fwrite(79,8,10,'╗');
[SSF]  456:   fwrite(65,9,10,'│'); fwrite(69,8,112,' Da
       tum '); fwrite(79,9,10,'│');
[CYD]  457:   fwrite(65,10,10,'╙'); PrChar(66,10,10,'='
       ,13); fwrite(79,10,10,'╜');
[025]  458:   fwrite(67,9,2,Datum);
[1L3]  459: (*  Uhrzeit  *)
```

Wir stellen vor

The TWIN Classic

Das kompatible Programm zu Lotus 1-2-3

Deutsche Menüs und deutsches Handbuch * äußerst flexible Gestaltung der Präsentationsgrafiken über Eingabemasken * Übernahme von 1-2-3-Dateien * Kein Kopierschutz * Kompatible zu den 1-2-3 Macros (englische Version) * leichte Einarbeitung durch ein ausführliches Handbuch

PC-MANAGEMENT:

"Clones fordern LOTUS heraus"
Der erste Clone von LOTUS 1-2-3 ist erschienen und wird unter der Behauptung "unglaubliche Ähnlichkeit zu 1-2-3" verkauft was auch stimmt.

PC-Magazine:

"Firmen, die momentan 1-2-3 anwenden, können sicher TWIN in dieselbe Umgebung einführen. Sowohl Rechenblatt- als auch Datenbankfunktionen sind ein 1-2-3 Clone. Das Programm bietet zu diesem Preis einen hervoragenden Gegenwert. Ein leistungsfähiges Spreadsheet mit sehr guter Grafik.

WHAT'S NEW IN COMPUTING:

TWIN bietet die Chance, die Leistungsfähigkeit von üblichen Programmen zu einem vernünftigen Preis zu erwerben.

30 Tage Geld-zurück-Garantie

JA, Sie können TWIN auf Ihrem IBM XT/AT, Schneider PC 1512 oder Kompatiblen (min 512 kB RAM) für 30 Tage testen. Wenn Sie feststellen, daß TWIN nicht das leistet, was wir behaupten und innerhalb 30 Tagen nach Kauf in gutem Zustand zurücksenden, erstatten wir Ihnen den Kaufpreis zurück.

Einführungsangebot

DM 199,90

Die Tabellenkalkulation TWIN V1.05
Ein Sparpreis aber kein Sparprogramm:

Lieferumfang:
drei 5.25"-Disketten oder
drei 3.5"-Disketten
1 Handbuch + Registrierungskarte

Eigenschaften:
- lauffähig auf dem 8086, 80286 und 80386
- einfache Konfigurierung
- Unterstützung der 8087- und 80287-Coprozessoren
- kein Kopierschutz
- 8192 Zeilen, 256 Spalten
- Einlesen und Schreiben von 1-2-3-Dateien
- Einlesen und Schreiben von DIF-Dateien
- Einlesen von Visicalc-Dateien
- Anzeige des verfügbaren Speicherplatzes
- frei wählbare Farben für Arbeitsblatt und Menüs
- Unterstützung von CGA und HGC für die Grafiken
- 8 Diagrammgrundarten
- Auswahl unter 11 verschiedenen Schriftarten für Überschriften und Beschriftungen
- logarithmische Skalen
- zwei Y-Achsen
- Grafiken speicher- und abrufbar

MOSAIC
SOFTWARE INC
0721-848955

Ja, ich bin beeindruckt ! Bitte schicken Sie mir ein TWIN Classic zum Einführungspreis von DM 199,90. Ich lege einen Scheck/ Verrechnungsscheck über DM 199,90 bei.

Name _____
Funktion _____
Straße _____
Plz/Ort _____
Telefon _____

Adresse: MOSAIC SOFTWARE INC.
Lessingstr. 39, 7500 Karlsruhe 1

MOSAIC und TWIN sind eingetragene Warenzeichen von MOSAIC SOFTWARE INC. Andere Produktnamen sind Warenzeichen der jeweiligen Hersteller.

```
[ZYD]  460:   fwrite(65,12,10,'╔'); PrChar(66,12,10,'='
           ,13); fwrite(79,12,10,'╗');
[YTH]  461:   fwrite(65,13,10,'║'); fwrite(68,12,112,'
           Uhrzeit '); fwrite(79,13,10,'║');
[GZD]  462:   fwrite(65,14,10,'╚'); PrChar(66,14,10,'='
           ,13); fwrite(79,14,10,'╝');
[XA6]  463:   fwrite(70,13,2,Uhrzeit);
[TUO]  464: end;
[000]  465:
[7YA]  466: (*****************************************
           ********************************)
[5XD]  467: (*                    Laufwerks-Fehler anzei
           gen                              *)
[7YA]  468: (*****************************************
           ********************************)
[000]  469:
[UOC]  470: procedure  DiskFehler( LW : char; F : byte
           );
[000]  471:
[BH1]  472: begin
[WW1]  473:   case F of
[YOH]  474:    2  : Message('Diskette im Laufwerk '+LW+
           ': nicht formatiert'+
[ZA4]  475:                ' oder defekt !');
[XYH]  476:    128: Message('Laufwerk '+LW+': nicht ber
           eit oder keine Diskette'+
[A24]  477:                ' eingelegt !');
[TUO]  478:   end;
[TUO]  479: end;
[000]  480:
[7YA]  481: (*****************************************
           ********************************)
[VMK]  482: (*       Einlesen des Pfades vom Quell- und
           ZielLaufwerk                     *)
[7YA]  483: (*****************************************
           ********************************)
[000]  484:
[2H9]  485: procedure  LiesPfad( LW : char );
[000]  486:
[BH1]  487: begin
[8MA]  488:   if LW=QLw then getdir(ord(LW)-64,QPfad)
[R28]  489:     else getdir(ord(LW)-64,ZPfad);
[PO8]  490:   if QLw=ZLw then ZPfad:=QPfad
[TUO]  491: end;
[000]  492:
[7YA]  493: (*****************************************
           ********************************)
[Z7L]  494: (*        Überprüfen des Pfades vom Quell-
           und Ziellaufwerk                 *)
[7YA]  495: (*****************************************
           ********************************)
[000]  496:
[UKE]  497: function  TestPfad( LW : char; TPfad : Pfad
           ): boolean;
[000]  498:
[BH1]  499: begin
[KO7]  500:   (*$I-*) chdir(LW+':'+TPfad); (*$I+*)
[MVA]  501:   if IOresult <> 0 then TestPfad:= false
[A11]  502:   else
[BH1]  503:   begin
[Y75]  504:     TestPfad:=true;
[X83]  505:     LiesPfad(LW)
[TUO]  506:   end;
[TUO]  507: end;
[000]  508:
[7YA]  509: (*****************************************
           ********************************)
[LBG]  510: (*             Q U I C K  - Sort zum Sortie
           ren des Directory's              *)
[7YA]  511: (*****************************************
           ********************************)
[000]  512:
[MZ9]  513: procedure  QSort( Von, Bis : Index );
[000]  514:
[EMO]  515: var
[MH4]  516:       x,w : DirEintrag;
[5F2]  517:       i,k : Index;
[000]  518:
[BH1]  519: begin
[2K4]  520:   i:=Von; k:=Bis;
[NA8]  521:   x:=Directory(.(Von + Bis) div 2.);
[X52]  522:   repeat
[X9E]  523:     while Directory(.i.).FName < x.FName do
           i:=succ(i);
[UGE]  524:     while x.FName < Directory(.k.).FName do
           k:=pred(k);
[OT3]  525:     if i <= k then
[BH1]  526:     begin
[MO5]  527:       w:=Directory(.i.);
[129]  528:       Directory(.i.):=Directory(.k.);
[ES4]  529:       Directory(.k.):=w;
[WO3]  530:       i:=succ(i);
[7V2]  531:       k:=pred(k)
[DLO]  532:     end
[R53]  533:   until i > k;
[648]  534:   if Von < k then QSort(Von,k);
[SH7]  535:   if i < Bis then QSort(i,Bis)
[TUO]  536: end;
[000]  537:
[7YA]  538: (*****************************************
           ********************************)
[79H]  539: (*          Anzeige des Directory's in eine
           m Fenster => F10                 *)
[7YA]  540: (*****************************************
           ********************************)
[000]  541:
[8SK]  542: procedure  ZeigeDir( LW : char; TPfad : Pfa
           d );   Label Ausgang;
[000]  543:
[EMO]  544: var
[054]  545:       x,z,y     : byte;
[GF3]  546:       Ein       : char;
[DY4]  547:       Temp      : FileName;
[YT3]  548:       Str1      : Str80;
[000]  549:
[BH1]  550: begin
[6AE]  551:   FensterAuf(' Directory von Laufwerk '+LW+
           ': ',6,8,75,22);
[FNB]  552:   fwrite(6,8,2,'Name im Laufwerk '+LW+': is
           t ==>  ');
[5E6]  553:   fwrite(35,8,10,Ausg); z:=11;
[KAC]  554:   fwrite(6,9,2,'Verzeichnis im Laufwerk ist
           ');
[R95]  555:   fwrite(35,9,10,TPfad);
[F75]  556:   for x:=1 to BZahl do
[BH1]  557:   begin
[LY3]  558:     for y:=0 to 4 do
[BH1]  559:     begin
[OP9]  560:       if x+y > BZahl then goto Ausgang;
[NU2]  561:       if z=20 then
[BH1]  562:       begin
[TXF]  563:         fwrite(18,22,10,'weitere Auflistung
           mit einer Taste');
[PY2]  564:         CursorAus;
[X52]  565:         repeat
[ZR3]  566:           read(Kbd,Ein);
[6DA]  567:           if Ein=#27 then if keypressed the
           n
[BH1]  568:           begin
[ZR3]  569:             read(Kbd,Ein);
[1D6]  570:             if Ein=#59 then Help1;
[AN1]  571:             Ein:=#27
[DLO]  572:           end
[A11]  573:           else
[BH1]  574:           begin
[PY2]  575:             CursorAus;
[H63]  576:             FensterZu;
[O31]  577:             exit
[TUO]  578:           end;
[265]  579:         until not (Ein=#27);
[Y83]  580:         clrscr; z:=9
[TUO]  581:       end;
[3R8]  582:       if BildName(.y+x.).SubDir then
[ZY9]  583:         fwrite(6+(y*14),z,10,BildName(.y+x.
           ).BName)
[A11]  584:       else
[SRA]  585:         fwrite(6+(y*14),z,2,BildName(.y+x.)
           .BName);
[TUO]  586:     end;
[GN4]  587:     z:=succ(z); x:=x+4
[TUO]  588:   end;
[AV2]  589: Ausgang:
[FEG]  590:   if LW=QLw then str(QMax:9:0,Str1) else st
           r(ZMax:9:0,Str1);
[AZE]  591:   fwrite(6,21,2,Str1+' Bytes sind maximal v
           erfügbar');
[RZF]  592:   if LW=QLw then str(QFrei:9:0,Str1) else s
           tr(ZFrei:9:0,Str1);
[UNB]  593:   fwrite(6,22,2,Str1+' Bytes sind noch frei
           ');
[PY2]  594:   CursorAus;
[X52]  595:   repeat
[ZR3]  596:     read(Kbd,Ein);
[6DA]  597:     if Ein=#27 then if keypressed then
[BH1]  598:     begin
[ZR3]  599:       read(Kbd,Ein);
[1D6]  600:       if Ein=#59 then Help1;
[TUO]  601:     end;
[EW5]  602:   until Ein in (.#13,#27,#32.);
[V23]  603:   FensterZu
[TUO]  604: end;
[000]  605:
[7YA]  606: (*****************************************
           ********************************)
[DFF]  607: (*                Umwandlung von KBytes in M
           Bytes                            *)
[7YA]  608: (*****************************************
           ********************************)
[000]  609:
[P5B]  610: function  KB_oder_MB( Max : real ) : KB_MB;
[000]  611:
```

```
[EMO]  612: var
[VE2]  613:        KB     : real;
[773]  614:        Temp   : KB_MB;
[000]  615:
[BH1]  616: begin
[PA3]  617:   KB:= Max / 1024.0;
[5T3]  618:   if KB > 2047 then
[BH1]  619:   begin
[KO3]  620:     KB:= KB / 1024.0;
[AH4]  621:     str(KB:4:0,Temp);
[195]  622:     insert(' MB',Temp,5);
[365]  623:     KB_oder_MB:=Temp;
[DLO]  624:   end
[A11]  625:   else
[BH1]  626:   begin
[AH4]  627:     str(KB:4:0,Temp);
[X85]  628:     insert(' KB',Temp,5);
[365]  629:     KB_oder_MB:=Temp;
[TUO]  630:   end;
[TUO]  631: end;
[000]  632:
[7YA]  633: (*****************************************
       ********************************)
[3VK]  634: (*          Berechnung der KBytes vom Quel
       l- und Ziellaufwerk             *)
[7YA]  635: (*****************************************
       ********************************)
[000]  636:
[DE9]  637: procedure  BerechneKB( LW : char );
[000]  638:
[EMO]  639: var
[T61]  640:        Frei,
[3T2]  641:        Max    : real;
[V44]  642:        KB     : string(.6.);
[000]  643:
[BH1]  644: begin
[F13]  645:   with REGS do
[BH1]  646:   begin
[ZM8]  647:     DL:=ord(LW)-64; AH:=$36; MsDos(REGS);
[WN3]  648:     if AX > 0 then
[BH1]  649:     begin
[EV4]  650:       Frei:= 1.0 * AX * CX * BX;
[8L4]  651:       Max := 1.0 * AX * CX * DX;
[X24]  652:       if LW=QLw then
[BH1]  653:       begin
[Y13]  654:         QFrei:=Frei;
[RC2]  655:         QMax :=Max;
[WB6]  656:         QKB  :=KB_oder_MB(QMax);
[DLO]  657:       end
[A11]  658:       else
[BH1]  659:       begin
[723]  660:         ZFrei:=Frei;
[OD2]  661:         ZMax :=Max;
[OD6]  662:         ZKB  :=KB_oder_MB(ZMax);
[TUO]  663:       end;
[LS3]  664:       LwFlag:=true;
[DLO]  665:     end
[8A5]  666:     else LwFlag:=false;
[TUO]  667:   end;
[TUO]  668: end;
[000]  669:
[7YA]  670: (*****************************************
       ********************************)
[VMM]  671: (*        Anzeige der Kapazität (Bytes) vom
       Quell- und Ziellaufwerk         *)
[7YA]  672: (*****************************************
       ********************************)
[000]  673:
[OZ7]  674: procedure  Space( LW : char );
[000]  675:
[EMO]  676: var
[E73]  677:        Ein          : char;
[VF2]  678:        Gesamt,
[LE2]  679:        Belegt,
[OU6]  680:        Verfuegbar   : string(.10.);
[000]  681:
[BH1]  682: begin
[LLE]  683:   FensterAuf(' Kapazität von Laufwerk '+LW+
       ': ',43,14,76,22);
[PY2]  684:   CursorAus;
[6YD]  685:   fwrite(42,21,10,'╟'); PrChar(43,21,10,'='
       ,34); fwrite(77,21,10,'╢');
[MG4]  686:   BerechneKB(LW);
[X24]  687:   if LW=QLw then
[BH1]  688:   begin
[LE5]  689:     str(QMax:10:0,Gesamt);
[ZF7]  690:     str(QMax-QFrei:10:0,Belegt);
[KV6]  691:     str(QFrei:10:0,Verfuegbar)
[DLO]  692:   end
[A11]  693:   else
[BH1]  694:   begin
[OG5]  695:     str(ZMax:10:0,Gesamt);
[YH7]  696:     str(ZMax-ZFrei:10:0,Belegt);
[ZW6]  697:     str(ZFrei:10:0,Verfuegbar)
[TUO]  698:   end;
```

```
[D1B]  699:   fwrite(47,15,2,'Gesamt:     '+Gesamt+' Byt
       es');
[7ZA]  700:   fwrite(47,17,2,'Belegt:     '+Belegt+' Byt
       es');
[NAE]  701:   fwrite(47,19,2,'Verfügbar: '+Verfuegbar+'
       Bytes');
[86C]  702:   fwrite(49,22,10,'Weiter mit einer Taste')
       ;
[X52]  703:   repeat
[PY2]  704:     CursorAus;
[ZR3]  705:     read(Kbd,Ein);
[6DA]  706:     if Ein=#27 then if keypressed then
[BH1]  707:     begin
[ZR3]  708:       read(Kbd,Ein);
[BD6]  709:       if Ein=#59 then Help3;
[TUO]  710:     end;
[EW5]  711:   until Ein in (.#13,#27,#32.);
[H63]  712:   FensterZu;
[XA6]  713:   fwrite(70,13,2,Uhrzeit)
[TUO]  714: end;
[000]  715:
[7YA]  716: (*****************************************
       ********************************)
[U8D]  717: (*                    Einlesen des Director
       y's                          *)
[7YA]  718: (*****************************************
       ********************************)
[000]  719:
[D1B]  720: procedure  LiesDirectory( LW : char );
[000]  721:
[EMO]  722: var
[VC3]  723:          x        : byte;
[MP4]  724:          TName    : FileName;
[9S5]  725:          Maske    : string(.4.);
[206]  726:          TPfad    : string(.64.);
[UV3]  727:          DTA      : record
[D69]  728:                       Reserviert : array(.1..
       21.) of byte;
[R45]  729:                       Attribut   : byte;
[OW4]  730:                       Zeit       : integer;
[885]  731:                       Datum      : integer;
[TA9]  732:                       Bytes      : array(.1..
       2.) of integer;
[AG7]  733:                       Name       : array(.1..
       13.) of char;
[TUO]  734:                     end;
[000]  735:
[BH1]  736: begin
[G5F]  737:   AnZahl:= 1; BZahl:=1; fillchar(DTA,128,'
       '); Maske:='*.*'+chr(0);
[F13]  738:   with REGS do
[BH1]  739:   begin
[VK8]  740:     AH:=$0E; DX:=ord(LW)-65; MsDos(REGS);
[HLB]  741:     AH:=$1A; DS:=seg(DTA); DX:=ofs(DTA); Ms
       Dos(REGS);
[KUG]  742:     AH:=$4E; DS:=seg(Maske(.1.)); DX:=ofs(M
       aske(.1.)); CX:=$08; MsDos(REGS);
[NM4]  743:     TName:=DTA.Name;
[HZ9]  744:     if TName(.9.)='.' then delete(TName,9,1
       );
[EZ7]  745:     for x:=1 to length(TName) do
[PSG]  746:       if ((TName(.x.)='.') and (TName(.x+1.
       )='.')) then delete(TName,x,1);
[K23]  747:     Ausg:=TName;
[BND]  748:     if TName='                ' then Ausg:='ke
       in Name vorhanden';
[856]  749:     fillchar(DTA.Name,14,' ');
[5X6]  750:     AH:=$4E; CX:=$11; MsDos(REGS);
[OD6]  751:     while ((Flags and 1) = 0) do
[BH1]  752:     begin
[XV7]  753:       with Directory(.AnZahl.) do
[BH1]  754:       begin
[GB7]  755:         if DTA.Attribut=16 then
[BH1]  756:         begin
[4V9]  757:           BildName(.BZahl.).BName:=DTA.Name
       ;
[T3A]  758:           BildName(.BZahl.).SubDir:=true;
[HK5]  759:           AnZahl:=pred(AnZahl)
[DLO]  760:         end
[A11]  761:         else
[BH1]  762:         begin
[7M4]  763:           FName:=DTA.Name;
[5M4]  764:           DName:=DTA.Name;
[4V9]  765:           BildName(.BZahl.).BName:=DTA.Name
       ;
[SEA]  766:           BildName(.Bzahl.).SubDir:=false;
[1F6]  767:           if pos('.',DName) > 0 then
[LAE]  768:             while (pos('.',DName) < 9) do i
       nsert(' ',DName,pos('.',DName));
[NM5]  769:           Markiert:=false;
[LE5]  770:           if DTA.Bytes(.1.) >= 0 then
[N3B]  771:             FLaenge:= FFFF * DTA.Bytes(.2.)
       + DTA.Bytes(.1.)
[A11]  772:           else
[5WC]  773:             FLaenge:= FFFF * DTA.Bytes(.2.)
       + FFFF + DTA.Bytes(.1.)
```

```
[TUO]  774:          end;
[856]  775:          fillchar(DTA.Name,14,' ');
[RV5]  776:          AnZahl:=succ(AnZahl);
[BN4]  777:          BZahl:=succ(BZahl)
[TUO]  778:        end;
[HE4]  779:        AH:=$4F; MsDos(REGS)
[TUO]  780:      end;
[SV5]  781:      AnZahl:=pred(AnZahl);
[OV4]  782:      BZahl:=pred(BZahl);
[TUO]  783:    end;
[NF4]  784:    QSort(1,AnZahl)
[TUO]  785: end;
[000]  786:
[7YA]  787: (*****************************************
       ******************************)
[DZF]  788: (*           Eingabe und Setzen der Pfade -
        Taste F9                    *)
[7YA]  789: (*****************************************
       ******************************)
[000]  790:
[67C]  791: procedure  Pfade_eingeben;   Label Ende;
[000]  792:
[EMO]  793: var
[A73]  794:        TPfad : Pfad;
[WS2]  795:        Ein   : char;
[000]  796:
[BH1]  797: begin
[EPA]  798:   FensterAuf(' Pfade eingeben ',13,10,69,14
       );
[KYD]  799:   fwrite(12,13,10,'╠'); PrChar(13,13,10,'='
       ,57); fwrite(70,13,10,'╣');
[TZA]  800:   fwrite(14,11,2,QLw+':'); fwrite(14,12,2,Z
      Lw+':');
[X52]  801:   repeat
[TF5]  802:     gotoxy(1,5); clreol;
[LF5]  803:     gotoxy(5,2); clreol;
[XB8]  804:     TPfad:=Input(5,2,52); CursorAus;
[7B8]  805:     if TPfad(.1.)=#27 then goto Ende;
[7F6]  806:     if pos(' ',TPfad) > 0 then
[5ME]  807:       while (pos(' ',TPfad) > 0) do delete(
      TPfad,pos(' ',TPfad),1);
[G3B]  808:     if (TPfad <> '') and (length(TPfad) >=
       1) then
[BH1]  809:       begin
[YH7]  810:         if TestPfad(QLw,TPfad) then
[BH1]  811:         begin
[VA8]  812:           if (QLw+':'+TPfad) <> QPfad then
[BH1]  813:           begin
[SC3]  814:             QPfad:=TPfad;
[DLO]  815:           end
[DLO]  816:         end
[A11]  817:         else
[BH1]  818:         begin
[KL8]  819:           sound(2000); delay(250); nosound;
[RUG]  820:           fwrite(25,14,138,'Pfad im Laufwerk
      '+QLw+': nicht vorhanden !');
[1E2]  821:           delay(5000)
[TUO]  822:         end;
[TF5]  823:         gotoxy(1,5); clreol;
[YMM]  824:         fwrite(15,14,10,'Nochmalige Eingabe m
       it einer Taste, weiter mit ENTER');
[PY2]  825:         CursorAus;
[XC3]  826:         read(Kbd,Ein)
[DLO]  827:       end
[9C3]  828:       else Ein:=#13;
[6C3]  829:     until Ein=#13;
[X52]  830:     repeat
[TF5]  831:       gotoxy(1,5); clreol;
[NF5]  832:       gotoxy(5,3); clreol;
[XB8]  833:       TPfad:=Input(5,3,52); CursorAus;
[7B8]  834:       if TPfad(.1.)=#27 then goto Ende;
[7F6]  835:       if pos(' ',TPfad) > 0 then
[5ME]  836:         while (pos(' ',TPfad) > 0) do delete(
      TPfad,pos(' ',TPfad),1);
[G3B]  837:       if (TPfad <> '') and (length(TPfad) >=
       1) then
[BH1]  838:       begin
[BL7]  839:         if TestPfad(ZLw,TPfad) then
[BH1]  840:         begin
[AFC]  841:           if (ZLw+':'+ZPfad) <> TPfad then ZP
      fad:=TPfad;
[DLO]  842:         end
[A11]  843:         else
[BH1]  844:         begin
[KL8]  845:           sound(2000); delay(250); nosound;
[DWG]  846:           fwrite(25,14,138,'Pfad im Laufwerk
      '+ZLw+': nicht vorhanden !');
[1E2]  847:           delay(5000)
[TUO]  848:         end;
[TF5]  849:         gotoxy(1,5); clreol;
[YMM]  850:         fwrite(15,14,10,'Nochmalige Eingabe m
       it einer Taste, weiter mit ENTER');
[PY2]  851:         CursorAus;
[XC3]  852:         read(Kbd,Ein)
```

```
[DLO]  853:       end
[9C3]  854:       else Ein:=#13;
[6C3]  855:     until Ein=#13;
[381]  856: Ende:
[V23]  857:   FensterZu
[TUO]  858: end;
[000]  859:
[7YA]  860: (*****************************************
       ******************************)
[TD9]  861: (*                                M E N Ü  - 1 -
                                            *)
[7YA]  862: (*****************************************
       ******************************)
[000]  863:
[WS5]  864: procedure  Menue_1;
[000]  865:
[2X3]  866: var    z : byte;
[000]  867:
[BH1]  868: begin
[RXA]  869:   for z:=3 to 21 do fwrite(0,z,2,'
                ');
[KKA]  870:   for z:=16 to 21 do fwrite(65,z,2,'
                ');
[E25]  871:   fwrite(67,5,2,'           ');
[GV3]  872:   TAnz:= 1; CNr:=1;
[ZFL]  873:   fwrite(24,10,2,chr(24)+' '+chr(25)+'   -
       Quell- und Ziellaufwerk ändern  ');
[Z7F]  874:   fwrite(24,12,2,'F10   - Directory des Que
       11-Laufwerkes  ');
[VLC]  875:   fwrite(24,14,2,'F9    - Pfadnamen eingebe
      n                ');
[A9E]  876:   fwrite(24,16,2,'ENTER - Weiter zum nächst
      en Menü         ');
[71B]  877:   fwrite(24,18,2,'ESC   - Zurück zu DOS
                ');
[H58]  878:   fwrite(24,20,2,'
                ');
[XA6]  879:   fwrite(70,13,2,Uhrzeit);
[TUO]  880: end;
[000]  881:
[7YA]  882: (*****************************************
       ******************************)
[UCE]  883: (*                        Anzeigen der markierte
      n Bytes                       *)
[7YA]  884: (*****************************************
       ******************************)
[000]  885:
[397]  886: procedure  ZeigeBytes;
[000]  887:
[BH1]  888: begin
[P29]  889:   str(KB:8:0,Bytes); str(MK:8,Mark);
[TC5]  890:   fwrite(68,17,10,Mark);
[TM5]  891:   fwrite(68,20,10,Bytes)
[TUO]  892: end;
[000]  893:
[7YA]  894: (*****************************************
       ******************************)
[RD9]  895: (*                                M E N Ü  - 2 -
                                            *)
[7YA]  896: (*****************************************
       ******************************)
[000]  897:
[YS5]  898: procedure  Menue_2;
[000]  899:
[EMO]  900: var
[DG2]  901:       x : byte;
[000]  902:
[BH1]  903: begin
[YPE]  904:   fwrite(0,3,10,'╒'); PrChar(1,3,10,'=',18)
       ; fwrite(19,3,10,'╕');
[N69]  905:   fwrite(4,3,112,' Auswahl '+QLw+': ');
[RN3]  906:   for x:=4 to 20 do
[BH1]  907:   begin
[OL4]  908:     fwrite( 0,x,10,'│');
[XB4]  909:     fwrite(19,x,10,'│')
[TUO]  910:   end;
[VLF]  911:   fwrite(0,21,10,'╘'); PrChar(1,21,10,'=',1
       8); fwrite(19,21,10,'╛');
[30E]  912:   fwrite(65,16,10,'╒'); PrChar(66,16,10,'='
       ,13); fwrite(79,16,10,'╕');
[Z2H]  913:   fwrite(65,17,10,'│'); fwrite(67,16,112,'
       Markierte '); fwrite(79,17,10,'│');
[MOE]  914:   fwrite(65,18,10,'╘'); PrChar(66,18,10,'='
       ,13); fwrite(79,18,10,'╛');
[YOE]  915:   fwrite(65,19,10,'╒'); PrChar(66,19,10,'='
       ,13); fwrite(79,19,10,'╕');
[24H]  916:   fwrite(65,20,10,'│'); fwrite(69,19,112,'
       Bytes '); fwrite(79,20,10,'│');
[VYD]  917:   fwrite(65,21,10,'╘'); PrChar(66,21,10,'='
       ,13); fwrite(79,21,10,'╛');
[RX5]  918:   KB:=0; MK:=0; ZeigeBytes;
[LT5]  919:   fwrite(67,4,2,QLw+': '+QKB);
[NV5]  920:   fwrite(67,5,2,ZLw+': '+ZKB);
[CNL]  921:   fwrite(24,10,2,chr(24)+' '+chr(25)+'
       - Einen Dateinamen vor / zurück  ');
```

```
[WWE]  922:   fwrite(24,12,2,'PgUp/Dn - Eine Seite vor
       / zurück       ');
[TEC]  923:   fwrite(24,14,2,'F10     - Weitere Optione
       n              ');
[GKF]  924:   fwrite(24,16,2,'SPACE   - Datei zum Kopie
       ren markieren  ');
[80F]  925:   fwrite(24,18,2,'ENTER   - Markierte Datei
       en kopieren    ');
[1VD]  926:   fwrite(24,20,2,'ESC     - Zurück zum letz
       ten Menü       ');
[XA6]  927:   fwrite(70,13,2,Uhrzeit);
[ET3]  928:   TAnz:=1; CNr:=1;
[TUO]  929: end;
[000]  930:
[7YA]  931: (*****************************************
       ********************************)
[S1L]  932: (*            Überprüfung der Zieldateien
       auf dem Ziellaufwerk             *)
[7YA]  933: (*****************************************
       ********************************)
[000]  934:
[5V8]  935: function  TestFile : boolean;
[000]  936:
[BH1]  937: begin
[E15]  938:   assign( ZF, ZFile );
[P25]  939:   (*$I-*) reset( ZF );  (*$I+*)
[YNG]  940:   if IOresult = 0 then TestFile:=true else
       TestFile:=false;
[Y82]  941:   close( ZF )
[TUO]  942: end;
[000]  943:
[7YA]  944: (*****************************************
       ********************************)
[2V9]  945: (*                          K O P I E R E N
                                        *)
[7YA]  946: (*****************************************
       ********************************)
[000]  947:
[Y76]  948: procedure  Kopieren;
[000]  949:
[EM0]  950: var
[EB2]  951:       Ein : char;
[000]  952:
[BH1]  953: begin
[078]  954:   if QPfad(.4.) in (.'0'..'9','A'..'Z'.) th
       en
[47A]  955:     QFile:=QPfad+'\'+Directory(.z.).FName
[EEB]  956:   else QFile:=QPfad+Directory(.z.).FName;
[488]  957:   if ZPfad(.4.) in (.'0'..'9','A'..'Z'.) th
       en
[D7A]  958:     ZFile:=ZPfad+'\'+Directory(.z.).FName
[HHB]  959:   else ZFile:=ZPfad+Directory(.z.).FName;
[395]  960:   if TestFile then
[BH1]  961:   begin
[MLH]  962:     Message('Datei bereits vorhanden !. Übe
       rschreiben (J/N) ?');
[X52]  963:     repeat
[ZR3]  964:       read(Kbd,Ein);
[MN6]  965:     until Ein in (.'j','J','n','N'.);
[5S7]  966:     if not (upcase(Ein)='J') then
[BH1]  967:     begin
[NF3]  968:       Message(' ');
[031]  969:       exit
[TUO]  970:     end;
[TUO]  971:   end;
[NF3]  972:   Message(' ');
[YKB]  973:   fwrite(30,23,138,'Kopiervorgang läuft !')
       ;
[6B8]  974:   assign(QF,QFile); reset(QF);
[W29]  975:   assign(ZF,ZFile); rewrite(ZF);
[E8A]  976:   RestRec:=filesize(QF); Nr:=addr(QF);
[1F9]  977:   REGS.AX:=$5700; REGS.BX:=Nr^; MsDos(REGS)
       ;
[765]  978:   while RestRec > 0 do
[BH1]  979:   begin
[MFC]  980:     if MaxRec < RestRec then AnzRec:=MaxRec
       else AnzRec:=RestRec;
[S3A]  982:     blockread(QF,BufferPtr^,AnzRec);
[VDA]  983:     blockwrite(ZF,BufferPtr^,AnzRec);
[F28]  984:     RestRec:=RestRec-AnzRec;
[TUO]  985:   end;
[6A3]  986:   Nr:=addr(ZF);
[7F9]  987:   REGS.AX:=$5701; REGS.BX:=Nr^; MsDos(REGS)
       ;
[0W5]  988:   close(ZF); close(QF);
[TUO]  989: end;
[000]  990:
[7YA]  991: (*****************************************
       ********************************)
[PVK]  992: (*      Ausgabe der Dateien im Auswahl-Fens
       ter zum Markieren                *)
[7YA]  993: (*****************************************
       ********************************)
[000]  994:
```

```
[7Z5]  995: procedure  Auswahl;
[000]  996:
[EM0]  997: var
[EG2]  998:       y : byte;
[K83]  999:       d : integer;
[000] 1000:
[BH1] 1001: begin
[U64] 1002:   d:=pred(TAnz);
[WN3] 1003:   for y:=5 to 20 do
[UR6] 1004:     with Directory(.TAnz.) do
[BH1] 1005:     begin
[CU6] 1006:       if TAnz <= AnZahl then
[BH1] 1007:       begin
[375] 1008:         if Markiert then
[GTD] 1009:           if TAnz = d+CNr then fwrite(3,y,1
      12,chr(16)+DName)
[K39] 1010:           else fwrite(3,y,2,chr(16)+DName)
[A11] 1011:         else
[35C] 1012:           if TAnz = d+CNr then fwrite(3,y,1
      12,' '+DName)
[U17] 1013:           else fwrite(3,y,2,' '+DName);
[XY4] 1014:         TAnz:=succ(TAnz)
[DLO] 1015:       end
[TZ6] 1016:       else fwrite(3,y,2,'              ');
[TUO] 1017:     end;
[1G3] 1018:   TAnz:=TAnz-16;
[107] 1019:   if TAnz < 1 then TAnz:=1;
[TUO] 1020: end;
[000] 1021:
[7YA] 1022: (*****************************************
       ********************************)
[UOP] 1023: (*      Anzeige der markierten Dateien im A
      uswahlfenster beim Kopieren     *)
[7YA] 1024: (*****************************************
       ********************************)
[000] 1025:
[1C9] 1026: procedure  Anzeigen;   Label Loop;
[000] 1027:
[EM0] 1028: var
[S02] 1029:       y   : byte;
[HE3] 1030:       M   : integer;
[000] 1031:
[BH1] 1032: begin
[741] 1033:   M:=z;
[DR3] 1034:   for y:=5 to 19 do
[BH1] 1035:   begin
[TA1] 1036: Loop:
[RG4] 1037:     if M <= AnZahl then
[BH1] 1038:     begin
[11A] 1039:       if Directory(.M.).Markiert then
[BH1] 1040:       begin
[HE9] 1041:         fwrite(4,y,10,Directory(.M.).DName)
      ;
[D67] 1042:         fwrite(4,succ(y),2,'             ');
[4Z2] 1043:         M:=succ(M);
[DLO] 1044:       end
[A11] 1045:       else
[BH1] 1046:       begin
[4Z2] 1047:         M:=succ(M);
[YH2] 1048:         goto Loop
[TUO] 1049:       end;
[TUO] 1050:     end;
[TUO] 1051:   end;
[TUO] 1052: end;
[000] 1053:
[7YA] 1054: (*****************************************
       ********************************)
[S7E] 1055: (*                  Markierte Dateien löschen
                                        *)
[7YA] 1056: (*****************************************
       ********************************)
[000] 1057:
[V68] 1058: procedure  LoescheFiles;
[000] 1059:
[EM0] 1060: var
[CE3] 1061:       z      : Index;
[924] 1062:       JaNein : char;
[000] 1063:
[BH1] 1064: begin
[PY2] 1065:   CursorAus;
[NEG] 1066:   Message('Markierte Dateien wirklich lösch
      en (J/N) ?');
[X52] 1067:   repeat
[EZ4] 1068:     read(Kbd,JaNein);
[1Y8] 1069:   until upcase(JaNein) in (.'J','N'.);
[W47] 1070:   if upcase(JaNein)='J' then
[BH1] 1071:   begin
[2S5] 1072:     for z:=1 to AnZahl do
[6H6] 1073:     with Directory(.z.) do
[BH1] 1074:     begin
[375] 1075:       if Markiert then
[BH1] 1076:       begin
[TX4] 1077:         assign( QF, FName );
[282] 1078:         erase( QF )
[DLO] 1079:       end
```

```
[TUO] 1080:      end;
[TUO] 1081:    end;
[E63] 1082:    Message(' ')
[TUO] 1083:  end;
[000] 1084:
[7YA] 1085:  (*****************************************
        *********************************)
[APF] 1086:  (*                  Suchstring für Dateien e
        rstellen                          *)
[7YA] 1087:  (*****************************************
        *********************************)
[000] 1088:
[U4B] 1089:  procedure  SuchName( var Name : Str12 );
[000] 1090:
[EMO] 1091:  var
[CD1] 1092:         P : 0..12;
[982] 1093:         Z : char;
[000] 1094:
[BH1] 1095:  begin
[6R4] 1096:    P:=pos('.',TName);
[92F] 1097:    if P=0 then begin TName:=TName+'.'; P:=le
        ngth(TName) end;
[GU7] 1098:    Name:=copy(TName,1,pred(P));
[NA9] 1099:    if Name(.length(Name).)='*' then
[BH1] 1100:    begin
[7LA] 1101:      Name:=copy(Name,1,pred(length(Name)));
[621] 1102:      Z:='?'
[XW3] 1103:    end else Z:=' ';
[W1E] 1104:    while length(Name) < 8 do Name:=Name+Z; N
        ame:=Name+'.';
[N9D] 1105:    Name:=Name+copy(TName,succ(P),length(TNam
        e)-P);
[NA9] 1106:    if Name(.length(Name).)='*' then
[BH1] 1107:    begin
[7LA] 1108:      Name:=copy(Name,1,pred(length(Name)));
[621] 1109:      Z:='?'
[XW3] 1110:    end else Z:=' ';
[PXA] 1111:    while length(Name) < 12 do Name:=Name+Z
[TUO] 1112:  end;
[000] 1113:
[7YA] 1114:  (*****************************************
        *********************************)
[Z6F] 1115:  (*                    Speziellen Dateinamen e
        ingeben                           *)
[7YA] 1116:  (*****************************************
        *********************************)
[000] 1117:
[YV7] 1118:  procedure  DNameEingabe;
[000] 1119:
[EMO] 1120:  var
[WL4] 1121:        JaNein   : char;
[CC3] 1122:        i        : byte;
[3X1] 1123:        passt,
[UR4] 1124:        MFlag    : boolean;
[000] 1125:
[BH1] 1126:  begin
[FNE] 1127:    FensterAuf(' Dateinamen spezifizieren ',2
        5,10,57,13);
[KH7] 1128:    CursorAus; MFLag:=false;
[VXD] 1129:    fwrite(24,12,10,'╠'); PrChar(25,12,10,'='
        ,33); fwrite(58,12,10,'╣');
[CY5] 1130:    fwrite(27,11,2,QLw+': ==> ');
[X52] 1131:    repeat
[VDB] 1132:      gotoxy(1,4); clreol; gotoxy(11,2); clre
        ol;
[X45] 1133:      TName:=Input(11,2,12);
[BN5] 1134:      if TName(.1.)=#27 then
[BH1] 1135:      begin
[H63] 1136:        FensterZu;
[031] 1137:        exit
[TUO] 1138:      end;
[LG6] 1139:      if pos(' ',TName) > 0 then
[DPE] 1140:        while (pos(' ',TName) > 0) do delete(
        TName,pos(' ',TName),1);
[Z87] 1141:      if length(TName) > 0 then
[BH1] 1142:      begin
[U1G] 1143:        for i:=1 to length(TName) do TName(.i
        .):=upcase(TName(.i.));
[ZB8] 1144:        SuchName( Name ); CursorAus;
[8MC] 1145:        fwrite(28,13,10,'Eingabe korrigieren
        (J/N) ?');
[X52] 1146:        repeat
[EZ4] 1147:          read(Kbd,JaNein);
[528] 1148:        until JaNein in (.'j','J','n','N'.);
[B26] 1149:      end else JaNein:='N';
[UT6] 1150:    until upcase(JaNein)='N';
[Z87] 1151:    if length(TName) > 0 then
[2S5] 1152:    for z:=1 to AnZahl do
[BH1] 1153:    begin
[T83] 1154:      passt:=true;
[LL3] 1155:      for i:=1 to 12 do
[DWE] 1156:        if (Directory(.z.).DName(.i.)<>Name(.
        i.)) and (Name(.i.)<>'?')
[295] 1157:          then passt:=false;
[B84] 1158:      if passt then
[ZWB] 1159:        if Directory(.z.).Markiert=false then
```

```
[BH1] 1160:        begin
[648] 1161:          KB:=KB+Directory(.z.).FLaenge;
[5D3] 1162:          MK:=succ(MK);
[ED9] 1163:          Directory(.z.).Markiert:=true;
[PT2] 1164:          MFlag:=true
[TUO] 1165:        end;
[TUO] 1166:    end;
[BW4] 1167:    if not MFlag then
[BH1] 1168:    begin
[6TE] 1169:      Message('Keine Dateien zum markieren ge
        funden !');
[VN2] 1170:      delay(7000);
[E63] 1171:      Message(' ')
[TUO] 1172:    end;
[PY2] 1173:    CursorAus;
[V23] 1174:    FensterZu
[TUO] 1175:  end;
[000] 1176:
[7YA] 1177:  (*****************************************
        *********************************)
[29C] 1178:  (*                              Initialisierung
                                          *)
[7YA] 1179:  (*****************************************
        *********************************)
[000] 1180:
[2D5] 1181:  procedure  Init;
[000] 1182:
[EMO] 1183:  var
[WD6] 1184:        Befehl   : string(.127.);
[000] 1185:
[BH1] 1186:  begin
[NM2] 1187:    clrscr;
[Y66] 1188:    fillchar(ZPfad,64,' ');
[C66] 1189:    fillchar(QPfad,64,' ');
[5R4] 1190:    new(BufferPtr);
[NR6] 1191:    if ParamCount > 1 then
[BH1] 1192:    begin
[MS5] 1193:      Befehl:=ParamStr(1);
[2SC] 1194:      if Befehl(.2.)=':' then QLw:=upcase(Bef
        ehl(.1.));
[PS5] 1195:      Befehl:=ParamStr(2);
[GPC] 1196:      if Befehl(.2.)=':' then ZLw:=upcase(Bef
        ehl(.1.))
[TUO] 1197:    end;
[F13] 1198:    with REGS do
[BH1] 1199:    begin
[BM4] 1200:      AH:=$19; MsDos(REGS);
[YP2] 1201:      ADrive:=AL
[TUO] 1202:    end;
[LC2] 1203:    Test_GA;
[39E] 1204:    if not (QLw in (.'A'..'D'.)) or not (ZLw
        in (.'A'..'D'.)) then
[BH1] 1205:    begin
[AL5] 1206:      QLw:=chr(ADrive+65);
[A32] 1207:      ZLw:=QLw
[TUO] 1208:    end;
[7U4] 1209:    BerechneKB(QLw);
[G42] 1210:    Screen
[TUO] 1211:  end;
[000] 1212:
[7YA] 1213:  (*****************************************
        *********************************)
[Z1A] 1214:  (*                        H A U P T P R O G R
        A M M                             *)
[7YA] 1215:  (*****************************************
        *********************************)
[000] 1216:
[PW2] 1217:  Label Menu1;
[000] 1218:
[BH1] 1219:  begin
[RA1] 1220:    Init;
[6M1] 1221:  Menu1:
[X52] 1222:    repeat
[ZD2] 1223:      Menue_1;
[X52] 1224:      repeat
[PY2] 1225:        CursorAus;
[635] 1226:        read(Kbd,Eingabe);
[Y8A] 1227:        if (Eingabe=#27) and keypressed then
[BH1] 1228:        begin
[635] 1229:          read(Kbd,Eingabe);
[M26] 1230:          case ord(Eingabe) of
[RB4] 1231:           59: begin       (* F1 *)
[NF3] 1232:                 Message(' ');
[DL1] 1233:                 Help1;
[XA6] 1234:                 fwrite(70,13,2,Uhrzeit)
[TUO] 1235:               end;
[GA2] 1236:           72: begin
[NF3] 1237:                 Message(' ');
[W56] 1238:                 if QLw > 'C' then QLw:='A'
[Z08] 1239:                   else QLw:=chr(succ(ord(QLw
        )));
[674] 1240:                 fwrite(46,6,10,QLw)
[TUO] 1241:               end;
[DA2] 1242:           80: begin
[NF3] 1243:                 Message(' ');
```

ProSoft-Preise liegen richtig!

☎ 0261/4047-1 · Tx 862476 PSOFT · Telefax 0261/4047-252

Wir suchen ständig günstige Einkaufsquellen für die angebotenen und neue innovative Produkte.
Günstige Möglichkeit der Finanzierung durch Ratenkredit. Fordern Sie die Unterlagen an.

Commodore-PC — Commodore-PC

NEU! Commodore PC-10 III
PC-10 III Doppelfloppy-Version 8088-2 mit 4,77/7,16 und 9.54 MHz Taktfrequenz, 640 KB Hauptspeicher, Auto-Configuration BIOS, parallele und serielle Schnittstelle, Maus-Interface, AGA-Grafikadapter (Monochrom und Farbe), Echtzeituhr mit Kalenderfunktion, 2 Diskettenlaufw. à 360 KB, MF-Tastatur mit sep. Cursorblock, 12"-Monitor, MS-DOS 3.2 und GW-Basic **1798,–**

PC-10 III 2/20 Festplatten-Version 20 MB wie PC-10 III, jedoch mit 20 MB Festplatte zusätzlich **2398,–**

PC-10 III 2/30 Festplatten-Version 30 MB wie PC-10 III, jedoch mit 30 MB Festplatte zusätzlich **2498,–**

Das Profi-System! PC-10 III 2/50 Festplatten-Version 50 MB wie PC-10 III, jedoch mit 50 MB Festplatte zusätzlich **2798,–**

PC-20 III wie PC-10 III, jedoch nur 1 Diskettenlaufwerk 360 KB und 20 MB Festplatte **2598,–**

NEU! Commodore 386 PC-60/40
Commodore PC 60/40 80386 CPU mit 4,77/6/8/10/12/16 MHz Takt umschaltbar 1 MB Hauptspeicher, 2 serielle und 2 parallele Schnittstellen, EGA-Grafikadapter, 1 Diskettenlaufwerk 1.2 MB, 1 Festplatte 40 MB, MF-Tastatur mit sep. Cursorblock, Monitor 14", MS-DOS 3.2 und GW-Basic **9696,–**

Commodore PC 60/122 wie PC 60/40 jedoch mit 122 MB Festplatte anstatt 40 MB Festplatte **11498,–**

Amiga 2000 mit Monitor 1084 2796,–
Amiga 500 1148,–

Seagate — Seagate — Seagate

ST-225	20 MB, 65 ms, halbe Bauhöhe	448,–
ST-238R	30 MB, 65 ms, RRL-fähig, halbe Bauhöhe	468,–
ST-251-0	40 MB, 40 ms, halbe Bauhöhe	788,–
ST-251-1	40 MB, 28 ms, halbe Bauhöhe	938,–
ST-277R	65 MB, 40 ms, RLL-fähig, halbe Bauhöhe	878,–
ST-4096	80 MB, 40 ms, volle Bauhöhe	1598,–
ST-4144R	122 MB, 28 ms, RLL-fähig, volle Bauhöhe	1898,–

20 MB Festplattenkit ST-225 incl. XT-Controller u. Kabelsatz **588,–**
30 MB Festplattenkit ST-238 incl. XT-RLL-Controller u. Kabelsatz **618,–**

3,5" Festplattenlaufwerke

ST-125-0	20 MB, 40 ms, halbe Bauhöhe	528,–
ST-125-1	20 MB, 28 ms, halbe Bauhöhe	619,–
ST-157R-0	50 MB, 40 ms, halbe Bauhöhe	818,–
ST-157R-1	50 MB, 28 ms, halbe Bauhöhe	898,–

Tandon TM-9252 10 MB Festplatte **228,–**

Microscience — Microscience

HH-825	20 MB, 65 ms, halbe Bauhöhe	478,–
HH-830	30 MB, 65 ms, RLL-fähig, halbe Bauhöhe	528,–
HH-1050	40 MB, 28 ms, halbe Bauhöhe	1198,–
HH-1060	66 MB, 28 ms, RLL-fähig, halbe Bauhöhe	1298,–

3,5" Festplattenlaufwerke

HH-325	20 MB, 80 ms, halbe Bauhöhe	528,–
HH-330	30 MB, 80 ms, RLL-fähig, halbe Bauhöhe	538,–

Priam V-185 110 MB, 18 ms, RLL-fähig **1698,–**

Hardcards 20 MB Hardcard **648,–** 40 MB Hardcard **1698,–**

30 MB Hardcard (Microscience) **698,–**

Controller

XT Controller	148,–
XT-RLL Controller	188,–
AT Controller	298,–
AT-RLL Controller	398,–

Formatierungssoftware

Disk Manager (Ontrack)	19,–
SpeedStor	118,–
V-Feature de Luxe	238,–

Diskettenlaufwerke
360 KB Diskettenlaufwerk 198,– 1,2 MB Diskettenlaufwerk 298,–

10 MB Diskettenlaufwerk (Subsystem)
Take Ten Diskettenlaufwerk 10 MB intern **1448,–**
Take Ten Diskettenlaufwerk 10 MB extern **1798,–**

Tape Streamer 40 MB „Alloy APT-40" jetzt Quick-Tape kompatibel incl. DC-2000 Cassette nur **748,–**
52 MB „Wangtek FAD 5000" für XT oder AT **858,–**
Zubehör: Datencassette DC-2000 (für APT-40) **68,–**
Datencassette DC-600A (für FAD 5000) **78,–**

Grafik-Adapter

NEU! EGA-Wonder Enhanced nur **448,–**

VEGA de Luxe Autoswitch	598,–
Hercules Graphic Card plus incl. RAM-FONT	578,–
Hercules kompatible Grafik-Karte	148,–
Paradise EGA	348,–
Paradise EGA Autoswitch 80-Zeichen (650 x 350)	368,–
Paradise EGA Autoswitch 132-Zeichen (650 x 480)	398,–
Genoa Super HI-RES	498,–

Monitore

NEU! Mitsubishi Autoscan EUM-1471 A **1298,–**
NEC JC 1401 P3E Multisync, 14" EGA Monitor 1298,–
NEC Multisync plus Paradise EGA Autoswitch 132-Zeichen **1648,–**
14" ADI-kompatibler Monitor grün oder bernstein 298,–

Co-Prozessoren – Intel – Intel – Intel

8087 (5 MHz)	218,–	8087 (8 MHz)	329,–
8087 (10 MHz)	499,–	80287 (6 MHz)	348,–
80287 (8 MHz)	498,–	80287 (10 MHz)	598,–
Fast-Sockel 80287-8	148,–	Fast-Sockel 80287-10	148,–
80387-16	1098,–	80387-20	1998,–

NEC – NEC – NEC – NEC

V20-8 für 8088	27,–	V30-8 für 8086	29,–

Handy Scanner von Cameron **598,–**
Handy Scanner HS-1000 DF I 478,–
Microsoft kompatible Mouse, seriell 98,–
Alles Zubehör und Software rund um den PC zu sehr günstigen Preisen. Fordern Sie die Preisliste an!

Plantron – Plantron – Plantron

XT-Kompatible, PT-LC und PT-XT Konfigurationen

PT LC 4,77/8 MHz Taktfrequenz, 256 KB Hauptspeicher, 1 Diskettenlaufwerk 360 KB, Herkules kompatible Grafikkarte, parallele Schnittstelle, MF-Tastatur, MS-DOS 3.2 und PBasic **1245,–**

PT LC/20 wie PT LC, jedoch mit 20 MB Festplatte **1898,–**

PT LC/30 wie PT LC, jedoch mit 30 MB Festplatte **1948,–**

PT-XT 4,77/8 MHz Taktfrequenz, 256 KB Hauptspeicher, 2 Diskettenlaufwerke 360 KB, Herkules kompatible Grafikkarte mit paralleler Schnittstelle, MF-Karte mit Uhr, paralleler und serieller Schnittstelle sowie Game-Port, MF-Tastatur, MS-DOS 3.2 und PBasic **1688,–**

PT-XT 2/20 wie PT-XT, jedoch mit 20 MB Festplatte **2348,–**

PT XT 2/30 wie PT-XT, jedoch mit 30 MB Festplatte **2398,–**

AT-Kompatible, PT-ST und PT-AT Konfigurationen

PT-ST 80286 mit 6/8 MHz, 640 KB Hauptspeicher, Herkules kompatible Grafikkarte mit paralleler Schnittstelle, 1 Diskettenlaufwerk 1.2 MB, MF-Tastatur, MS-DOS 3.2/PBasic und umfangreiche Bedienungsanleitung **2198,–**

PT-AT wie PT-ST, jedoch mit zusätzlicher Multi I/O-Karte (die mit einer seriellen und parallelen Schnittstelle sowie Game Port bestückt ist), AT-FD/HD-Controller **2598,–**

PT AT/20 wie PT-AT, jedoch mit 20 MB Festplatte **3148,–**

PT-AT/30 wie PT-AT, jedoch mit 30 MB Festplatte **3298,–**

PT-AT/40 wie PT-AT, jedoch mit 40 MB Festplatte **3498,–**

PT-AT/80 wie PT-AT, jedoch mit 80 MB Festplatte **4098,–**

PT-AT/122 wie PT-AT, jedoch mit 122 MB Festplatte **4398,–**

PT-286 AT (Baby-AT) mit 64 MB Festplatte und Super EGA-Karte incl. 14"-Monitor (grün, schwarz-weiß oder bernstein) **3748,–**

AT-386 Kompatible, PT-HT Konfigurationen

PT-HT/2 80386 mit 16 MHz Taktfrequenz, 1 MB Hauptspeicher, 1 Diskettenlaufwerk 1.2 MB, 1 Festplatte 32 MB, serielle und parallele Schnittstelle, Hercules kompatible Grafikkarte, MF-Tastatur, MS-DOS 3.2 und PBasic **5698,–**

PT-HT wie PT-HT/2, jedoch mit Super EGA-Karte und 64 MB-Festplatte **6298,–**

PT-HT/80 wie PT-HT/2, jedoch mit Super EGA-Karte und 80 MB-Festplatte **6698,–**

PT-HT/122 wie PT-HT/2, jedoch mit Super EGA-Karte und 122 MB-Festplatte **7198,–**

Auf Wunsch der Fa. Plantron bestätigen wir, daß wir die günstigen Preise durch Selbstaufrüsten und Einbau der Festplatten und Karten durch unsere geschulten Techniker erreichen können. – Selbstverständlich gilt dies nicht nur für Plantron-Produkte.

Netzwerke

– einfach zu installieren – einfach zu nutzen – Preiswert wie nie zuvor!

Trans-NET – Trans-NET
Trans-Net Starter Kit zur kompl. Vernetzung von 2 PC's besteh. aus: 2 Netzwerkk., Kabel mit Abschlußwiderst. und Tab-Boxen, NET-Softw. (Betriebss.), Betriebsanl. **1698,–**

Trans-Net Netzwerkkarte	698,–
NOVELL-Emulator	398,–
NETBIOS-Emulator	398,–
Schüler-Lehrer-Monitor	398,–
BUS-Repeater	899,–

NET-Board – NET-Board
Mit Cache-Funktion, 2,5 MBit, voll kompatibel zu Netbios

NET-Board Einsteckkarte	798,–
Trunk-Cable (Verkabelung für 2 PC's	248,–
Betriebssystem OS 6.20 C für 3 Benutzer	548,–
StarHub (Verstärker/-Sternverteiler 4fach)	1448,–
Repeater (Verst./Einsteckk.)	898,–

KAYPRO – KAYPRO – KAYPRO – KAYPRO

Kaypro PC Mono Pack	2398,–	Kaypro PC 30 Mono Pack	2998,–
Kaypro 286 i Mono Pack	4398,–		

Kaypro 386 E 40 mit 40 MB-Festplatte und EGA-Wonder Graphic Adapter **10478,–**
Kaypro 386 E 110 mit 110 MB-Festplatte und EGA-Wonder Graphic-Adapter **11678,–**

Olivetti – Olivetti – Olivetti

M 240/55 G Doppelfloppy-Version 8086 mit 10 MHz, 640 KB Hauptspeicher, 2 Diskettenlaufwerke à 360 KB, serielle und parallele Schnittstelle, Echtzeituhr, OGC-Controller, Tastatur II, Monitor, MS-DOS 3.2 und GW-Basic **3298,–**

M 240/0520 G Festplatten-Version 20 MB wie M240/55G, jedoch mit 20 MB Festplatte **3898,–**

M 240/0530 G Festplatten-Version 30 MB wie M240/55G, jedoch mit 30 MB Festplatte **3948,–**

M 240/0550 G Festplatten-Version 50 MB wie M240/55G, jedoch mit 50 MB Festplatte **4198,–**

Natürlich liefern wir das gesamte Angebot des M240. Bitte erfragen Sie unsere EGA- und Positiv-Versionen!

M28 Einstiegskonfiguration 1 80286 mit 8 MHz, 1 MB Hauptspeicher, serielle und parallele Schnittstelle, 1 Diskettenlaufwerk 1.2 MB, 1 Festplatte 20 MB, Monitor, Tastatur, MS-DOS 3.2 **4398,–**

Bitte fordern Sie unser Datenblatt über die gesamten M28 Konfigurationen an!

Schneider PC – Schneider PC

PC 2640 MM 4488,– **PC 2640 EM 5288,–**
Geräte des Typs PC 2640 waren zur Zeit der Drucklegung dieser Anzeige an die Händler noch nicht ausgeliefert. Es können Lieferzeiten entstehen, rufen Sie daher bitte an.

PC 1640
PC 1640 CD/HD 20 3178,– PC 1640 ECD/HD 20 3748,–

Drucker
DMP 3160 548,– DMP 4000 848,–

LQ-3500 **798,–**

Atari – Atari – Atari

Atari 520 STM Tastatur, 512 KB RAM, 192 KB ROM, HF-Modulator, Floppy SF 354, ohne Monitor, Maus, Basic **848,–**

Atari 1040 STF Tastatur, 1024 KB RAM, 192 KB ROM, integrierte Floppy 720 KB, ohne Monitor, Maus, Basic **1165,–**

Monitor SM-124 **469,–**

Der Preishammer von ProSoft:
Atari 1040 STF, Tastatur, 1024 KB RAM, 192 KB ROM, integrierte Floppy 720 KB, Monochrom-Monitor SM 124, Maus, Basic **1498,–**

Brother – Brother – Brother

M-1109	469,–	M-1409	799,–	M-1509	948,–
M-1709	1148,–	M-2024 L+	1999,–	M-1724 L	1399,–

Star – Star – Star

				NX-15	948,–
NL-10	539,–	zusätzliches Interface für NL-10			78,–
ND-10	898,–	ND-15	1198,–	NR-10	1148,–
NR-15	1398,–	NB 24-10	1298,–	NB 24-15	1798,–
NB-15	2348,–	SR-10	798,–	EBZ f. NL-10	199,–

NX-1000 (wie LC-10) **448,–**

Sharp — Pocket-Computer — Sharp

PC-1246 S	94,–	PC-1280	268,–	PC-1460	275,–
PC-1248	123,–	PC-1360	344,–	PC-1475	275,–
PC-1260	206,–	PC-1421	240,–	PC-1500 A	344,–
PC-1262	275,–	PC-1425	275,–	PC-1600	690,–
PC-1270	115,–	PC-1450	219,–	PC-2500	549,–

EPSON – EPSON – EPSON

FX-800 918,– **FX-1000 1188,–**

LQ-500	848,–	LQ-850	1198,–	LQ-1050	1598,–
LQ-2500	2438,–	IX-800	1568,–	EX-800	1298,–
EX-1000	1628,–	LX-800	529,–	SQ-2500	3188,–
LX-800 VC/P	619,–	Laser GQ-3500			3978,–

Einzelblatteinzug LQ-850 298,– Einzelblatteinzug LQ-1050 398,–
Görlitz-Interface EPSON/C64 oder C128 198,–

OKI – OKI – OKI

Okimate 20 Farbdrucker mit Interface **498,–**

Die günstigen Preise für die gesamte OKI-Produktpalette incl. Laserline erfahren Sie bei uns am Telefon!

NEC – NEC – NEC – NEC – NEC

P 5	**1798,–**	**P 2200**	**898,–**
P 7	**1398,–**	**P 6**	**1098,–**
P 7 color	**1648,–**	**P 6 color**	**1448,–**
P 7 seriell	**1648,–**	**P 6 seriell**	**1398,–**
P 7 seriell color	**2298,–**	**P 6 seriell color**	**1848,–**
Pin-Feed-Tractor für P 6	109,–	Bidirektionaler Tractor für P 6	298,–
Pin-Feed-Tractor für P 7	228,–	Bidirektionaler Tractor für P 7	348,–
Cut-Sheet-Feeder für P 7			598,–

Kyocera-Laserdrucker — Kyocera-Laserdrucker

F-1000	**4999,–**	F-2200	**10698,–**
F-1200	**7498,–**	F-3000	**14898,–**

Citizen – Citizen – Citizen

Nun hat sich Citizen dem Preisniveau des NL-10 angepaßt:
LSP-120 D Commodore- oder Parallel-Schnittstelle **394,–**
LSP-100 **568,–**

Panasonic

KX-P 1081	478,–	KX-P 1082	628,–	KX-P 1083	848,–
KX-P 1592	1195,–	KX-P 1595	1598,–		

Seikosha – Seikosha – Seikosha

SP-180 A	449,–	SP-1200 AS	528,–	SP-180 VC	449,–
MP-1300 AI	1098,–	SP-1200 AI	528,–	MP-5300 AI	1378,–

SL-80 Ai **nur 768,–** **SL-80 VC** **768,–**

Einzelblatteinzug SL-80 348,–

Disketten No-Name

	10 Stück	50 Stück	100 Stück
3" Maxell	70,– DM	330,– DM	650,– DM
3½" 1 D	35,– DM	149,– DM	249,– DM
3½" 2 D	39,– DM	159,– DM	299,– DM
5¼" 2 D	19,– DM	59,– DM	89,– DM

5¼" 1 D	**100 Stück**	**79,– DM**
5¼" 1 D	**1000 Stück**	**698,– DM**
5¼" 2 D	**1000 Stück**	**699,– DM**

5¼" No-Name HD, 1,6 MB 10 Stück **48,– DM**

ProSoft GmbH

Filiale München Theresienstr. 56, 8000 München 2, Tel. 089/280 93 89, **direkt bei der technischen Hochschule.** Bitte beachten Sie, daß nicht ständig sämtl. Ware in unserer Filiale München vorrätig ist. **Rufen Sie an!**

Bogenstr. 51–53, Postf. 207, D-5400 Koblenz-Goldgrube, Tel. (0261) 4047-1, Telex 862476, Telefax (0261) 4047-252

Alle Preise zuzügl. 10,– DM Versandkosten pro Paket. Lieferung per Nachnahme oder Vorkassescheck – Versandkosten Ausland DM 40,– pro Paket. **Lassen Sie sich keinen Bären aufbinden! ProSoft liefert Original-Produkte der führenden Hersteller. Überzeugen Sie sich selbst durch Abholung der Ware in unseren Verkaufs- und Vorführräumen in Koblenz. Wir gewähren Ihnen bei Barzahlung (kein Scheck) 2 % Skonto auf alle Preise, was vielleicht schon zur Deckung Ihrer Reisekosten ausreicht. Einige unserer Vorlieferanten liefern Produkte ohne die Seriennummer des Herstellers. In diesem Fall übernehmen wir anstelle der Herstellergarantie die unbeschränkte gesetzliche Gewährleistung.** 03/88

```
[V76] 1244:                 if ZLw > 'C' then ZLw:='A'
[N28] 1245:                   else ZLw:=chr(succ(ord(ZLw
      )));
[974] 1246:                 fwrite(46,7,10,ZLw)
[TU0] 1247:               end;
[DE4] 1248:          68: begin       (* F10 *)
[NF3] 1249:                 Message(' ');
[TD5] 1250:                 Fehler:=TestLW(QLw);
[XV4] 1251:                 if Fehler=0 then
[BH1] 1252:                 begin
[D34] 1253:                   LiesPfad(QLw);
[ZK5] 1254:                   LiesDirectory(QLw);
[7U4] 1255:                   BerechneKB(QLw);
[B45] 1256:                   ZeigeDir(QLw,QPfad)
[DL0] 1257:                 end
[178] 1258:                 else DiskFehler(QLw,Fehler);
[XA6] 1259:                 fwrite(70,13,2,Uhrzeit);
[TU0] 1260:               end;
[8Y5] 1261:         113: begin       (* ALT - F10 *)
[NF3] 1262:                 Message(' ');
[TD5] 1263:                 Fehler:=TestLW(ZLw);
[XV4] 1264:                 if Fehler=0 then
[BH1] 1265:                 begin
[Z34] 1266:                   LiesPfad(ZLw);
[YM5] 1267:                   LiesDirectory(ZLw);
[BV4] 1268:                   BerechneKB(ZLw);
[L65] 1269:                   ZeigeDir(ZLw,ZPfad)
[DL0] 1270:                 end
[A78] 1271:                 else DiskFehler(ZLw,Fehler);
[XA6] 1272:                 fwrite(70,13,2,Uhrzeit);
[TU0] 1273:               end;
[GD4] 1274:          67: begin       (* F9 *)
[NF3] 1275:                 Message(' ');
[TD5] 1276:                 Fehler:=TestLW(QLw);
[XV4] 1277:                 if Fehler=0 then
[BH1] 1278:                 begin
[TD5] 1279:                   Fehler:=TestLW(ZLw);
[XV4] 1280:                   if Fehler=0 then
[BH1] 1281:                   begin
[275] 1282:                     Pfade_eingeben;
[ED5] 1283:                     LiesDirectory(QLw)
[DL0] 1284:                   end
[A78] 1285:                   else DiskFehler(ZLw,Fehler
      );
[DL0] 1286:                 end
[178] 1287:                 else DiskFehler(QLw,Fehler);
[XA6] 1288:                 fwrite(70,13,2,Uhrzeit);
[TU0] 1289:               end;
[TU0] 1290:           end;
[TU0] 1291:         end;
[AH4] 1292:         if Eingabe=#27 then
[BH1] 1293:         begin
[F16] 1294:           dispose(BufferPtr);
[F13] 1295:           with REGS do
[BH1] 1296:           begin
[154] 1297:             AH:=$0E; DL:=ADrive;
[GM2] 1298:             MsDos(REGS)
[TU0] 1299:           end;
[RZ3] 1300:           Window(1,1,80,25);
[NM2] 1301:           clrscr;
[Y11] 1302:           halt
[TU0] 1303:         end;
[DG4] 1304:         if Eingabe=#13 then
[BH1] 1305:         begin
[WW4] 1306:           if QLw=ZLw then
[BH1] 1307:           begin
[5LG] 1308:             Message('Quell- und Ziellaufwerk
      sind identisch.  -'+
[ZV8] 1309:                     '  Kopieren nicht möglich
      !');
[N63] 1310:             Eingabe:=#32
[DL0] 1311:           end
[A11] 1312:           else
[BH1] 1313:           begin
[TD5] 1314:             Fehler:=TestLW(QLw);
[XV4] 1315:             if Fehler=0 then
[BH1] 1316:             begin
[7U4] 1317:               BerechneKB(QLw);
[TD5] 1318:               Fehler:=TestLW(ZLw);
[7V9] 1319:               if Fehler=0 then BerechneKB(ZLw
      )
[A11] 1320:             else
[BH1] 1321:             begin
[C17] 1322:               DiskFehler(ZLw,Fehler);
[N63] 1323:               Eingabe:=#32
[TU0] 1324:             end;
[DL0] 1325:           end
[A11] 1326:           else
[BH1] 1327:           begin
[807] 1328:             DiskFehler(QLw,Fehler);
[N63] 1329:             Eingabe:=#32
[TU0] 1330:           end;
[TU0] 1331:         end;
[TU0] 1332:       end;
[WF4] 1333:     until Eingabe=#13;
```

```
[ZK5] 1334:     LiesDirectory(QLw);
[6E2] 1335:     Menue_2;
[9X2] 1336:     Auswahl;
[X52] 1337:     repeat
[PY2] 1338:       CursorAus;
[635] 1339:       read(Kbd,Eingabe);
[Y8A] 1340:       if (Eingabe=#27) and keypressed then
[BH1] 1341:       begin
[635] 1342:         read(Kbd,Eingabe);
[W3A] 1343:         if Eingabe in [.#16..#25,#30..#38,#
      44..#50.] then
[BH1] 1344:         begin
[M26] 1345:           case ord(Eingabe) of
[HV1] 1346:             16: C:='Q';
[TV1] 1347:             17: C:='W';
[9V1] 1348:             18: C:='E';
[SV1] 1349:             19: C:='R';
[9V1] 1350:             20: C:='T';
[HV1] 1351:             21: C:='Z';
[EV1] 1352:             22: C:='U';
[4V1] 1353:             23: C:='I';
[CV1] 1354:             24: C:='O';
[FV1] 1355:             25: C:='P';
[RU1] 1356:             30: C:='A';
[BV1] 1357:             31: C:='S';
[YU1] 1358:             32: C:='D';
[2V1] 1359:             33: C:='F';
[5V1] 1360:             34: C:='G';
[8V1] 1361:             35: C:='H';
[CV1] 1362:             36: C:='J';
[FV1] 1363:             37: C:='K';
[KV1] 1364:             38: C:='L';
[SV1] 1365:             44: C:='Y';
[TV1] 1366:             45: C:='X';
[6V1] 1367:             46: C:='C';
[VV1] 1368:             47: C:='V';
[9V1] 1369:             48: C:='B';
[RV1] 1370:             49: C:='N';
[5V1] 1371:             50: C:='M';
[TU0] 1372:           end;
[453] 1373:           Flag:=false;
[2Z5] 1374:           for z:=1 to Anzahl do
[BH1] 1375:           begin
[9BD] 1376:             if (Directory(.z.).DName(.1.)=C
      ) and not Flag then
[BH1] 1377:             begin
[EF2] 1378:               TAnz:=z;
[XM4] 1379:               if TAnz < 17 then
[BH1] 1380:               begin
[UT2] 1381:                 TAnz:=z-15;
[1Z6] 1382:                 If TAnz < 1 then TAnz:=1;
[N63] 1383:                 CNr:=z mod 16;
[T56] 1384:                 if CNr = 0 then CNr:=16;
[1U5] 1385:                 if CNr < 1 then CNr:=1
[DL0] 1386:               end
[A11] 1387:               else
[BH1] 1388:               begin
[1M6] 1389:                 if TAnz+15 >= Anzahl then
[BH1] 1390:                 begin
[XZ3] 1391:                   TAnz:=Anzahl-15;
[ZP4] 1392:                   CNr:=16-(Anzahl-z)
[DL0] 1393:                 end
[A11] 1394:                 else
[BH1] 1395:                 begin
[EF2] 1396:                   TAnz:=z;
[MB1] 1397:                   CNr:=1
[TU0] 1398:                 end;
[TU0] 1399:               end;
[9X2] 1400:               Auswahl;
[SP2] 1401:               Flag:=true
[TU0] 1402:             end;
[TU0] 1403:           end;
[AV4] 1404:           if not Flag then
[BH1] 1405:           begin
[1N2] 1406:             sound(2000);
[ME2] 1407:             delay(250);
[UZ2] 1408:             nosound
[TU0] 1409:           end;
[DL0] 1410:         end
[A11] 1411:         else
[M26] 1412:         case ord(Eingabe) of
[WA2] 1413:          59: begin
[KL1] 1414:                Help2;
[XA6] 1415:                fwrite(70,13,2,Uhrzeit)
[TU0] 1416:              end;
[1CA] 1417:          72: begin   (* Cursor eine Zeile r
      auf *)
[DE4] 1418:               if CNr > 1 then
[BH1] 1419:               begin
[EC4] 1420:                 CNr:=pred(CNr);
[9X2] 1421:                 Auswahl
[DL0] 1422:               end
[A11] 1423:               else
[BH1] 1424:               begin
[HS1] 1425:                 CNr:=1;
```

```
[705] 1426:                   TAnz:=pred(TAnz);
[107] 1427:                   if TAnz < 1 then TAnz:=1;
[9X2] 1428:                   Auswahl
[TUO] 1429:                 end;
[TUO] 1430:               end;
[CBB] 1431:          80: begin   (* Cursor eine Zeile t
      iefer *)
[HZ6] 1432:                 if TAnz+CNr-1 < AnZahl then
[E64] 1433:                 if CNr < 16 then
[BH1] 1434:                 begin
[MD4] 1435:                   CNr:=succ(CNr);
[9X2] 1436:                   Auswahl
[DLO] 1437:                 end
[A11] 1438:                 else
[BH1] 1439:                 begin
[ER1] 1440:                   CNr:=16;
[SO5] 1441:                   TAnz:=succ(TAnz);
[R1A] 1442:                   if TAnz > AnZahl then TAnz
      :=AnZahl;
[9X2] 1443:                   Auswahl
[TUO] 1444:                 end;
[TUO] 1445:               end;
[ZD4] 1446:          73: begin   (* PgUp *)
[1U5] 1447:                 if CNr > 1 then CNr:=1
[A11] 1448:                 else
[BH1] 1449:                 begin
[1G3] 1450:                   TAnz:=TAnz-16;
[6Y6] 1451:                   if TAnz < 1 then TAnz:=1
[TUO] 1452:                 end;
[9X2] 1453:                 Auswahl
[TUO] 1454:               end;
[MC4] 1455:          81: begin   (* PgDn *)
[DO6] 1456:                 if CNr < 16 then CNr:=16
[A11] 1457:                 else
[BH1] 1458:                 begin
[VF3] 1459:                   TAnz:=TAnz+16;
[PNA] 1460:                   if TAnz >= AnZahl then TAn
      z:=AnZahl-15
[TUO] 1461:                 end;
[O3C] 1462:                 if TAnz+CNr >= AnZahl then T
      Anz:=AnZahl-15;
[US4] 1463:                 if TAnz < 1 then
[BH1] 1464:                 begin
[FZ1] 1465:                   TAnz:=1;
[BU2] 1466:                   CNr:=AnZahl
[TUO] 1467:                 end;
[9X2] 1468:                 Auswahl
[TUO] 1469:               end;
[MU4] 1470:          71: begin      (* HOME *)
[HS1] 1471:                 CNr:=1;
[FZ1] 1472:                 TAnz:=1;
[9X2] 1473:                 Auswahl
[TUO] 1474:               end;
[9L4] 1475:          79: begin      (* END *)
[ER1] 1476:                 CNr:=16;
[XY3] 1477:                 TAnz:=AnZahl-15;
[US4] 1478:                 if TAnz < 1 then
[BH1] 1479:                 begin
[FZ1] 1480:                   TAnz:=1;
[BU2] 1481:                   CNr:=AnZahl
[TUO] 1482:                 end;
[9X2] 1483:                 Auswahl
[TUO] 1484:               end;
[5W3] 1485:          68: begin   (* F10 *)
[NRC] 1486:                 FensterAuf(' Optionen ',22,4
      ,62,16); CursorAus;
[PUC] 1487:                 fwrite(23,5,2,'DEL - Markier
      te Dateien löschen');
[H4C] 1488:                 fwrite(23,7,2,' M  - Alle Da
      teien markieren');
[8FF] 1489:                 fwrite(23,9,2,' U  - Markier
      ung aller Dateien aufheben');
[XUE] 1490:                 fwrite(23,11,2,' N  - Spezie
      lle Dateinamen eingeben');
[FWE] 1491:                 fwrite(23,13,2,'F5     - Kap
      azität vom Laufwerk '+QLw+':');
[LDF] 1492:                 fwrite(23,15,2,'ALT F5 - Kap
      azität vom Laufwerk '+ZLw+':');
[X52] 1493:                 repeat
[ZR3] 1494:                   read(Kbd,Ein);
[6P9] 1495:                   if (Ein=#27) and keypresse
      d then
[BH1] 1496:                   begin
[ZR3] 1497:                     read(Kbd,Ein);
[5N4] 1498:                     case ord(Ein) of
[942] 1499:                      59 : Help3;
[MY3] 1500:                      63 : Space(QLw);
[WY3] 1501:                      108 : Space(ZLw);
[U62] 1502:                      83 : begin
[A54] 1503:                             LoescheFiles;
[ZK5] 1504:                             LiesDirectory(QL
      w);
[GC4] 1505:                             KB:=0; MK:=0; Ei
      n:=#32;
```

```
[B63] 1506:                             ZeigeBytes;
[ER1] 1507:                             CNr:=16;
[XY3] 1508:                             TAnz:=AnZahl-15;
[US4] 1509:                             if TAnz < 1 then
[BH1] 1510:                             begin
[FZ1] 1511:                               TAnz:=1;
[BU2] 1512:                               CNr:=AnZahl
[TUO] 1513:                             end;
[9X2] 1514:                             Auswahl;
[XA6] 1515:                             fwrite(70,13,2,U
      hrzeit);
[TUO] 1516:                           end;
[TUO] 1517:                       end;
[TUO] 1518:                     end;
[KK5] 1519:                     if Ein in (.'m','M'.) then
[BH1] 1520:                     begin
[361] 1521:                       KB:=0;
[2S5] 1522:                       for z:=1 to AnZahl do
[BH1] 1523:                       begin
[ED9] 1524:                         Directory(.z.).Markier
      t:=true;
[648] 1525:                         KB:=KB+Directory(.z.).
      FLaenge;
[Y33] 1526:                         MK:=succ(MK)
[TUO] 1527:                       end;
[B63] 1528:                       ZeigeBytes;
[CR1] 1529:                       Ein:=#32;
[9X2] 1530:                       Auswahl
[TUO] 1531:                     end;
[KM5] 1532:                     if Ein in (.'u','U'.) then
[BH1] 1533:                     begin
[G3F] 1534:                       for z:=1 to AnZahl do Di
      rectory(.z.).Markiert:=false;
[Z68] 1535:                       KB:=0; MK:=0; ZeigeBytes
      ; Ein:=#32;
[9X2] 1536:                       Auswahl
[TUO] 1537:                     end;
[UK5] 1538:                     if Ein in (.'n','N'.) then
[BH1] 1539:                     begin
[RT3] 1540:                       DNameEingabe;
[B63] 1541:                       ZeigeBytes;
[CR1] 1542:                       Ein:=#32;
[9X2] 1543:                       Auswahl;
[XA6] 1544:                       fwrite(70,13,2,Uhrzeit);
[TUO] 1545:                     end;
[EW5] 1546:                   until Ein in (.#13,#27,#32.)
      ;
[H63] 1547:                   FensterZu;
[XA6] 1548:                   fwrite(70,13,2,Uhrzeit);
[TUO] 1549:                 end;
[TUO] 1550:           end;
[DLO] 1551:         end
[A11] 1552:         else
[HG4] 1553:           if Eingabe=#32 then
[BH1] 1554:           begin
[SO9] 1555:             with Directory(.TAnz+CNr-1.) do
[BH1] 1556:             begin
[YS7] 1557:               if TAnz+CNr-1 <= AnZahl then
[BH1] 1558:               begin
[375] 1559:                 if Markiert then
[BH1] 1560:                 begin
[NM5] 1561:                   Markiert:=false;
[254] 1562:                   KB:=KB-FLaenge;
[KC3] 1563:                   MK:=pred(MK);
[MD4] 1564:                   CNr:=succ(CNr);
[T89] 1565:                   if CNr > AnZahl then CNr:=A
      nZahl;
[E64] 1566:                   if CNr > 16 then
[BH1] 1567:                   begin
[ER1] 1568:                     CNr:=16;
[SO5] 1569:                     TAnz:=succ(TAnz);
[FXB] 1570:                     if TAnz+CNr >= AnZahl the
      n TAnz:=AnZahl-15
[TUO] 1571:                   end;
[DLO] 1572:                 end
[A11] 1573:                 else
[BH1] 1574:                 begin
[PD5] 1575:                   Markiert:=true;
[M44] 1576:                   KB:=KB+FLaenge;
[5D3] 1577:                   MK:=succ(MK);
[MD4] 1578:                   CNr:=succ(CNr);
[T89] 1579:                   if CNr > AnZahl then CNr:=A
      nZahl;
[E64] 1580:                   if CNr > 16 then
[BH1] 1581:                   begin
[ER1] 1582:                     CNr:=16;
[SO5] 1583:                     TAnz:=succ(TAnz);
[FXB] 1584:                     if TAnz+CNr >= AnZahl the
      n TAnz:=AnZahl-15
[TUO] 1585:                   end;
[TUO] 1586:                 end;
[B63] 1587:                 ZeigeBytes;
[9X2] 1588:                 Auswahl
[TUO] 1589:               end;
```

```
[TU0] 1590:            end;
[DL0] 1591:          end
[A11] 1592:          else
[DG4] 1593:          if Eingabe=#13 then
[BH1] 1594:          begin
[7C4] 1595:            if KB > ZFrei then
[BH1] 1596:            begin
[3TC] 1597:              Message('Nicht genug Speicherpl
      atz auf '+
[E09] 1598:                              'Ziel-Disk/Platte vorha
      nden !');
[DS5] 1599:              delay(5000); Message(' ');
[BZA] 1600:              Message('Trotzdem kopieren (J/N
      ) ?');
[X52] 1601:              repeat
[LX4] 1602:                read(Kbd,JaNein)
[528] 1603:              until JaNein in ('j','J','n','
      N',);
[MZ6] 1604:              if upcase(JaNein) ='N' then
[BH1] 1605:              begin
[WF3] 1606:                Eingabe:=#32;
[E63] 1607:                Message(' ')
[DL0] 1608:              end
[A44] 1609:              else Message(' ');
[TU0] 1610:            end;
[TU0] 1611:          end;
[5L6] 1612:      until Eingabe in (.#13,#27.);
[LD6] 1613:    until not (Eingabe=#27);
[428] 1614:    LiesPfad(ZLw); LiesPfad(QLw);
[AW9] 1615:    for z:=5 to 20 do fwrite(3,z,2,'
       ');
[2S5] 1616:    for z:=1 to AnZahl do
[BH1] 1617:    begin
[WAA] 1618:      if Directory(.z.).Markiert then
[BH1] 1619:      begin
[XA6] 1620:        fwrite(70,13,2,Uhrzeit);
[MW2] 1621:        Anzeigen;
[5W2] 1622:        Kopieren
[TU0] 1623:      end;
[TU0] 1624:    end;
[NF3] 1625:    Message(' ');
[5L2] 1626:    goto Menu1
[5T0] 1627: end.
```

Listing 1. Der Quellcode für »scopy.pas«

```
[7YA]    0: (*****************************************
            *********************************)
[3VD]    1: (*                    Ausschaltung des Curso
            r's                              *)
[7YA]    2: (*****************************************
            *********************************)
[000]    3:
[2T6]    4: procedure  CursorAus;
[000]    5:
[EM0]    6: var
[D03]    7:       REGS : record
[6LA]    8:                AX,BX,CX,DX,BP,SI,DI,DS,IS,F
           lags : integer
[TU0]    9:              end;
[000]   10:
[BH1]   11: begin
[F13]   12:   with REGS do
[BH1]   13:   begin
[HF1]   14:     BX:= 0;
[XU2]   15:     DX:= 26 shl 8;
[MP2]   16:     DX:= DX or 1;
[672]   17:     AX:= 2 shl 8;
[GV2]   18:     INTR($10,REGS)
[TU0]   19:   end;
[TU0]   20: end;
[000]   21:
```

Listing 2. Include-Datei »cursor.inc« dient zum Abschalten des Cursors

```
[26B]    0: (*****************************************
            *********************************)
[S6B]    1: (*                                Fenster -  Modul
                                              *)
[26B]    2: (*****************************************
            *********************************)
[000]    3:
[VM1]    4: const
[LR7]    5:       BildsAdr       : integer=$B800;
[NR7]    6:       BildsI         : boolean=true;
[Z57]    7:       Test           : boolean=true;
[GS6]    8:       BildsPos       : integer=0;
[000]    9:
[A21]   10: type
```

```
[WA6]   11:       Str76          = string(.76.);
[000]   12:
[EM0]   13: var
[P3F]   14:       BildsGroesse   : array(.0..BildsBuff
            er.) of integer;
[LS6]   15:       AktFenster     : integer;
[YRB]   16:       FensterDaten   : array(.0..MaxFenste
            r.) of
[L52]   17:                        record
[YN5]   18:                          x1,y1,x2,y2
            : byte;
[9Z5]   19:                          Xpos,Ypos
            : byte;
[9N6]   20:                          BildsGrAdr
            : integer
[TU0]   21:                        end;
[000]   22:
[26B]   23: (*****************************************
            *********************************)
[CGP]   24: (*    Initialisierung von Variablen und tes
            ten des Bildschirm-Adapters      *)
[26B]   25: (*****************************************
            *********************************)
[000]   26:
[RU5]   27: procedure  Test_GA;
[000]   28:
[FB4]   29: var   REGS : record
[2LA]   30:                AX,BX,CX,DX,BP,SI,DI,DS,ES,Fl
            ags : integer
[TU0]   31:              end;
[000]   32:
[BH1]   33: begin
[U34]   34:   AktFenster:=0;
[LE6]   35:   FensterDaten(.0.).x1:=0;
[PE6]   36:   FensterDaten(.0.).y1:=0;
[3K6]   37:   FensterDaten(.0.).x2:=79;
[3H6]   38:   FensterDaten(.0.).y2:=24;
[BD3]   39:   REGS.AX:=$0F00;
[D83]   40:   INTR($10,REGS);
[T85]   41:   Test:=(REGS.AX and 255)=7;
[7R3]   42:   if Test then
[HZ3]   43:     BildsAdr:=$B000;
[4X3]   44:   BildsI:=false
[TU0]   45: end;
[000]   46:
[26B]   47: (*****************************************
            *********************************)
[4EA]   48: (*                      F E H L E R M E L D U
            N G E N                          *)
[26B]   49: (*****************************************
            *********************************)
[000]   50:
[7CB]   51: procedure  Error( Fehler : integer );
[000]   52:
[BH1]   53: begin
[SN6]   54:   Window(1,1,80,25); clrscr;
[TK4]   55:   write('Error ==> ');
[M54]   56:   case Fehler of
[NCC]   57:     1: writeln('Zuviele Fenster geöffnet');
[OBB]   58:     2: writeln('Fenster-Bufferüberlauf');
[KPE]   59:     3: writeln('Falsche Koordinaten des Fen
            sters');
[1AF]   60:     4: writeln('Überschrift des Fensters zu
            lang');
[TU0]   61:   end;
[Y11]   62:   halt
[TU0]   63: end;
[000]   64:
[26B]   65: (*****************************************
            *********************************)
[9EC]   66: (*                          Fenster öffnen
                                              *)
[26B]   67: (*****************************************
            *********************************)
[000]   68:
[16L]   69: procedure  FensterAuf( UeberSchrift : Str76
            ; x1,y1,x2,y2 : integer );
[000]   70:
[EM0]   71: var
[OY1]   72:       Len, z,
[GY5]   73:       Xpos, Ypos    : integer;
[000]   74:
[BH1]   75: begin
[9W6]   76:   if BildsI then Test_GA;
[GVD]   77:   if length(UeberSchrift) > x2-x1+1 then Er
            ror(4);
[WUC]   78:   if ( x2-x1 < 2 ) or ( y2-y1 < 2 ) or not(
            x1 in (.1..78.)) or
[UDA]   79:        not(x2 in (.1..78.)) or not(y1 in (.1
            ..23.)) or
[598]   80:        not(y2 in (.1..23.)) then Error(3);
[GMB]   81:   Xpos:= wherex; Ypos:= wherey; Window(1,1,
            80,25);
[YKE]   82:   if AktFenster=MaxFenster then Error(1); L
            en:=x2-x1+3;
```

```
[HOF]   83:    if Len*(y2-y1+2) > BildsBuffer-BildsPos t
          hen Error(2);
[K59]   84:    AktFenster:=succ(AktFenster);
[81A]   85:    FensterDaten(. AktFenster .).x1:=x1;
[G1A]   86:    FensterDaten(. AktFenster .).y1:=y1;
[K1A]   87:    FensterDaten(. AktFenster .).x2:=x2;
[U1A]   88:    FensterDaten(. AktFenster .).y2:=y2;
[NKB]   89:    FensterDaten(. AktFenster-1 .).Xpos:=Xpos
          ;
[VKB]   90:    FensterDaten(. AktFenster-1 .).Ypos:=Ypos
          ;
[DYE]   91:    FensterDaten(. AktFenster .).BildsGrAdr:=
          BildsPos;
[1B8]   92:    for z:=pred(y1) to succ(y2) do
[BH1]   93:    begin
[LKK]   94:      move( mem(.BildsAdr:z*160+pred(x1)*2.),
          BildsGroesse(.BildsPos.),Len*2 );
[LA7]   95:      BildsPos:=BildsPos + Len
[TUO]   96:    end;
[8U8]   97:    fwrite(pred(x1),pred(y1),10,'╔');
[80C]   98:    for z:=x1 to x2 do fwrite(z,pred(y1),10,'
          ═');
[PR8]   99:    fwrite(succ(x2),pred(y1),10,'╗');
[CU6]  100:    for z:=y1 to y2 do begin
[DU6]  101:                          fwrite(pred(x1),z,10
          ,'‖');
[TP6]  102:                          fwrite(succ(x2),z,10
          ,'‖')
[TUO]  103:                        end;
[8U8]  104:    fwrite(pred(x1),succ(y2),10,'╙');
[YOC]  105:    for z:=x1 to x2 do fwrite(z,succ(y2),10,'
          ═');
[2S8]  106:    fwrite(succ(x2),succ(y2),10,'╝');
[GNP]  107:    fwrite(x1+(x2-x1) div 2-length(UeberSchri
          ft) div 2+1,pred(y1),112,Ueberschrift);
[F9E]  108:    Window(succ(x1),succ(y1),succ(x2),succ(y2
          )); clrscr;
[TUO]  109: end;
[000]  110:
[7YA]  111: (*****************************************
          ********************************)
[BDF]  112: (*              Letztes Fenster wieder schli
          essen                          *)
[7YA]  113: (*****************************************
          ********************************)
[000]  114:
[KP6]  115: procedure  FensterZu;
[000]  116:
[XR2]  117: var    Len, z,
[UZ5]  118:        x1,y1,x2,y2  : integer;
[000]  119:
[BH1]  120: begin
[9F7]  121:    if AktFenster=0 then exit;
[X8A]  122:    x1:= FensterDaten(. AktFenster .).x1;
[19A]  123:    y1:= FensterDaten(. AktFenster .).y1;
[39A]  124:    x2:= FensterDaten(. AktFenster .).x2;
[79A]  125:    y2:= FensterDaten(. AktFenster .).y2;
[7BF]  126:    BildsPos:=FensterDaten(.AktFenster.).Bild
          sGrAdr;
[NZ2]  127:    Len:=x2-x1+3;
[1B8]  128:    for z:=pred(y1) to succ(y2) do
[BH1]  129:    begin
[UMK]  130:      move( BildsGroesse(.BildsPos.),mem(.Bil
          dsAdr:z*160+pred(x1)*2.),Len*2 );
[XW6]  131:      BildsPos:= BildsPos + Len
[TUO]  132:    end;
```

```
[50F]  133:    BildsPos:= FensterDaten(.AktFenster.).Bil
          dsGrAdr;
[Z49]  134:    AktFenster:=pred(AktFenster);
[X8A]  135:    x1:= FensterDaten(. AktFenster .).x1;
[19A]  136:    y1:= FensterDaten(. AktFenster .).y1;
[39A]  137:    x2:= FensterDaten(. AktFenster .).x2;
[79A]  138:    y2:= FensterDaten(. AktFenster .).y2;
[EYB]  139:    Window(succ(x1),succ(y1),succ(x2),succ(y2
          ));
[CDR]  140:    gotoxy( FensterDaten(.AktFenster.).Xpos ,
           FensterDaten(.AktFenster.).Ypos )
[TUO]  141: end;
```

Listing 3. Die Include-Datei »window.inc« enthält alle wichtigen Window-Routinen

```
[7YA]    0: (*****************************************
           ********************************)
[1PH]    1: (*            Routine zur schnelleren Bilds
           chirmausgabe                    *)
[7YA]    2: (*****************************************
           ********************************)
[000]    3:
[CGL]    4: procedure  fwrite( Zeile, Spalte, Attribut
           : byte; Str : Str80 );
[000]    5:
[BH1]    6: begin
[H52]    7:   inline
[91D]    8:      ($1E/$1E/$8A/$86/Spalte/$B3/$50/$F6/$E3
           /$2B/$DB/$8A/$9E/Zeile/
[YLA]    9:       $03/$C3/$03/$C0/$8B/$F8/$BE/$00/$00/$8
           A/$BE/Attribut/
[D3A]   10:       $8A/$8E/Str/$22/$C9/$74/$3E/$2B/$C0/$8
           E/$D8/$A0/$49/$04/
[W1A]   11:       $1F/$2C/$07/$74/$22/$BA/$00/$B8/$8E/$D
           A/$BA/$DA/$03/$46/
[VHA]   12:       $8A/$9A/Str/$EC/$A8/$01/$75/$FB/$FA/$E
           C/$A8/$01/$74/$FB/
[MS8]   13:       $89/$1D/$47/$47/$E2/$EA/$2A/$C0/$74/$1
           0/$BA/$00/$B0/
[ABA]   14:       $8E/$DA/$46/$8A/$9A/Str/$89/$1D/$47/$4
           7/$E2/$F5/$1F);
[TUO]   15: end;
```

Listing 4. Mit »fwrite.inc« wird die Bildschirmausgabe beschleunigt

```
[7YA]    0: (*****************************************
           ********************************)
[1RE]    1: (*                        Ausgabe einer Anzahl
           von Zeichen                     *)
[7YA]    2: (*****************************************
           ********************************)
[000]    3:
[1RF]    4: procedure  PrChar( x,y,F : byte; Z : char;
           Anz : byte );
[000]    5:
[CU3]    6: var      i : byte;
[000]    7:
[BH1]    8: begin
[5WB]    9:   for i:=0 to pred(Anz) do fwrite(x+i,y,F,Z
           )
[TUO]   10: end;
```

Listing 5. »prchar.inc« ist für die Bildschirmausgabe verantwortlich

WOERLTRONIC dataphon s 21 d

TECHNISCHE DATEN:

Stromversorgung
9 Volt Blockbatterie
9 Volt Akku
Ext. Netzteil 9–15 V
40–45 mA

Schnittstelle
V. 24/RS 232,
Standard 25-Pin
nach ISO 2110

Übertragungsgeschwindigkeit
max. 300 bit/sek.

unverbindl. Preisempfehlung DM 249,–

WOERLTRONIC dataphon s21/23 d

TECHNISCHE DATEN:

Stromversorgung:
wie dataphon s 21 d

Schnittstelle
V. 24/RS 232,
Standard 25-Pin
nach ISO 2110

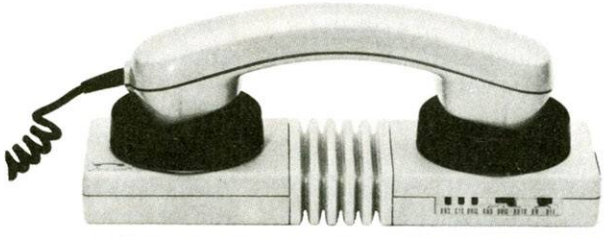
BTX-Rundbuchse

Übertragungsgeschwindigkeit
300–600–1200–
1200/75 bit/sek.

Autom. Geschwindigkeitsumsetzung
von 75 auf 1200 bit/sek.

unverbindl. Preisempfehlung DM 359,–

PROBLEMLOS MIT COMPUTERN IN ALLER WELT KOMMUNIZIEREN:
Per Telefon – von Datenbanken, Mailboxen, Großrechnern, zwischen Filialen, Zentralen, Außendiensten, Freunden und Bekannten – Daten, Texte und Programme abrufen oder übermitteln.



WOERLTRONIC – DATAPHON

Besuchen Sie uns auf der CeBit 88
Halle 4, 1.OG., Stand B 36
Laufende BTX-Vorführungen

Lieferbare Komplettsets bestehend aus:
WOERLTRONIC-dataphon
Anschlußkabel
Terminalprogramm
für:
APPLE-II e/+
ATARI-ST 260/520+
ATARI-XE/XL
COMMODORE C 64
IBM/Kombatible
SCHNEIDER 464
SCHNEIDER 664/6128

Info:
Wörlein GmbH+Co. KG
Postfach 4
D-8501 Cadolzburg
Tel. 09103/8294, Telex 625337

Erhältlich in Kaufhäusern, im Fachhandel und Versandhandel

Filed – der Editor, dem nichts verborgen bleibt

Mit Filed bekommen Sie einen Editor, der jedes Zeichen am Bildschirm darstellen kann. Die Utility arbeitet somit mit jeder Datei und schreckt selbst vor Exe- oder Com-Dateien nicht zurück.

Sicherlich standen Sie auch schon vor dem Problem, nur einen ganz kurzen Blick in eine Datei zu werfen, um dann resigniert aufzugeben, weil keines Ihrer Programme mit dieser Datei auch nur im entferntesten etwas anfangen kann. Der Type-Befehl gibt hämische Pfeiftöne von sich und zeigt zu allem Überfluß nur die Hälfte an. Auch der souveräne Griff nach einem teuren Textprogramm endet mit einer Enttäuschung. Auch diese müssen bei dieser speziellen Programmdatei passen und verweigern zum Teil die Arbeit völlig. Da hilft dann meist nur noch erneutes Booten.

Um die Utility Filed einzugeben, brauchen Sie Turbo Pascal 3.0 und einen CGA- oder Hercules-Monitor. Da direkt in den Bildschirmspeicher geschrieben wird, muß je nach Monitor-Art gesondert zugegriffen werden. Das vorliegende Listing ist für den CGA-Monitor (Farbgrafik) voreingestellt, die Werte für den Hercules-Bildschirm stehen jeweils in Klammern.

Das Programm erwartet als Parameter den Dateinamen der zu bearbeitenden Datei (eventuell mit Pfadnamen). Im nun sichtbaren Hauptmenü wurde auf viel »Schnick-Schnack« verzichtet (Bild 1). Filed stellt nur die wichtigsten Anzeigen zur Verfügung, damit vom Bildschirm soviel als möglich für die Anzeige der Datei frei bleibt. Das heißt natürlich nicht, daß Sie auf eine Help-Funktion verzichten müssen. Im Programm sind zwei Hilfe-Bildschirme integriert.

Am oberen Bildschirmrand informiert eine Anzeige über die Position des Cursor innerhalb der Datei. Wenn Sie mit den Cursortasten die Position des blinkenden Rechtecks verändern, ersehen Sie anhand des sich bewegenden Schrägstrichs in dieser Anzeige, wo Sie in der Datei arbeiten. Steht der Schrägstrich in der Mitte der Linie, ist die Hälfte der Datei geschafft. Das »A« steht für den Anfang, das »E« für das Ende der Datei.

In der untersten Zeile wird im linken Teil der Dateiname angezeigt. Zusätzlich gibt das Programm hier kurze Hilfsmeldungen aus. Etwas weiter rechts steht die Angabe der aktuellen Cursorposition in Byte (Pos:) und neben dem Trennstrich die effektive Länge der Datei. Die Meldung »Schreiben = Aus« auf der rechten Seite der untersten Zeile bedeutet, daß im Moment keine Änderungen an der Datei gemacht werden können.

Diese Vorsichtsmaßnahme ist nötig, da sonst schnell einmal aus Versehen auf eine Taste gedrückt wird und die überschriebenen Daten unter Umständen nicht ohne weiteres rekonstruiert werden können. Das vorherige Einschalten des »Schreibmodus« vermeidet derartige »Unfälle«.

Im eigentlichen Editierfeld erkennen Sie je nach Dateiart (Programm oder reiner ASCII-Text) mehr oder weniger sonderbare Zeichen. Dies sind die Sonderzeichen, die man mit dem einfachen Type-Befehl nicht sehen kann. Die vorgestellte Utility zeigt nun aber jedes Zeichen, das in der Datei steht (und sei es noch so eigenartig).

Vor dem tieferen Einstieg in das Programm befolgen Sie bitte folgenden Rat. Da die einzelnen Funktionen auf den

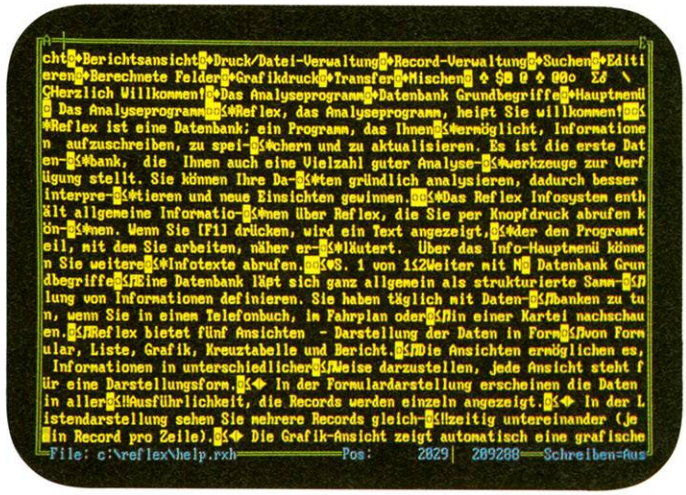


Bild 1. Der Editor zeigt jedes Sonderzeichen an

ersten Blick etwas schwierig erscheinen, ist die Anleitung so aufgebaut, daß die Eingabe Schritt für Schritt erklärt wird. Lesen Sie also grundsätzlich die komplette Erklärung zu einer Funktion, vielleicht sogar ein paar Sätze weiter, bis Sie sich wirklich über die Funktionsweise und die Bedienung im Klaren sind.

Schritt für Schritt werden Sie in die Bedienung von »Filed« eingeführt

Zunächst rufen Sie durch Drücken der F1-Taste das erste Hilfsmenü auf (Bild 2). Es bietet Ihnen eine Übersicht über die Steuerung des Cursors. Wenn Sie eine weitere Taste drücken, wird das zweite Hilfsmenü (Bild 3) ausgegeben. Es zeigt die Belegung der Funktions-Tasten. Mit einem weiteren Tastendruck gelangen Sie ins Hauptbild zurück.

Wenn Sie eine größere Datei aufgerufen haben, blättern Sie mit den Tasten [PgUp] und [PgDn]. Um vom Ende der Datei mit der Taste [Cursor-aufwärts] wieder an den Anfang zu gelangen, bleiben Sie einmal einfach mit dem Finger auf der Taste. Dabei wird nämlich nicht das ganze Bild mit neuen Daten überschrieben, sondern nur wenige Zeilen. Dieser Vorgang bricht erst nach Loslassen der Taste [Cursor-aufwärts] ab. Gleiches geschieht bei den Tasten [PgDn] und [PgUp]. Dieses Leistungsmerkmal ermöglicht auch bei »langsamen« PCs ein schnelles Blättern, ohne den Überblick zu verlieren.

Halten Sie die Ctrl-Taste gedrückt und drücken Sie dazu die Tasten [Home], [End], [PgDn] und [PgUp]. So steuern Sie innerhalb des Bildschirms alle vier Eckpositionen an. Innerhalb einer Zeile genügen die Tastenkombinationen [Ctrl-Cursor-links] und [Ctrl-Cursor-rechts], um an den Anfang beziehungsweise an das Ende der Zeile zu kommen. Mit den Tasten [Home] und [End] (ohne [Ctrl]) kommen Sie direkt an den Datei-Anfang beziehungsweise an das Datei-Ende. »Filed« kann im Hauptbildschirm jederzeit durch Betätigung der Taste [ESC] beendet werden.

Die Steuerung des Utilities übernehmen zum großen Teil die Funktionstasten

Der folgende Abschnitt erklärt ausführlich die Funktionstasten und ihre Wirkung im Editor.

Funktionstaste 1 [F1]: (Hilfsfunktion):
Wie bereits erwähnt, ruft die Taste [F1] die beiden Hilfsbildschirme (Bild 2 und Bild 3) auf. Die Hilfsanzeige können Sie jederzeit durch Drücken einer beliebigen Taste verlassen.

Funktionstaste 2 [F2]: (Neue Datei ansprechen)
Mit dieser Taste wählen Sie eine neue Datei für die Bearbeitung aus.

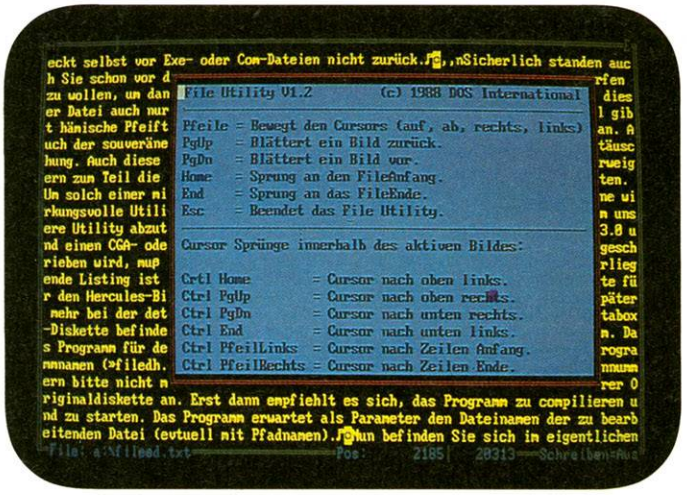


Bild 2. Hilfsbildschirm 1 zur Anzeige der Steuerfunktionen

Funktionstaste 3 [F3]: (Finde eine Zeichenfolge)
Sie werden nach einer Zeichenfolge gefragt, die dann in der Datei gesucht wird. Die Angabe »Länge:« bezieht sich auf die Länge der eingegebenen Zeichenfolge. Dies ist nützlich für die Funktion »Suchen und Ersetzen« (wird weiter unten erklärt).
Geben Sie nun einmal den Suchbegriff »abc« ein. Die Zeichenfolge schließen Sie mit [Enter] ab. Sie stellen dann wahrscheinlich mit Erstaunen fest, daß [Enter] umgewandelt und angezeigt, aber nicht ausgeführt wird. Das liegt daran, daß jede Taste, außer den Funktionstasten, den Cursor-Steuertasten und den Tasten [Ins] und [Del], als ganz normale Eingabetaste erkannt wird und zwar im selben Format, wie sie auch in einer Datei gespeichert ist. Um nun die eingegebene Zeichenfolge »abc« abzuschließen, müßten Sie anstelle der Enter-Taste die End-Taste benutzen.
Zunächst muß nun dieses »falsche« Enter-Zeichen gelöscht werden. Mit der Backspace-Taste finden Sie aber bestenfalls erneut ein »falsches« Zeichen im Eingabebefehl vor, denn diese Taste wird ebenfalls umgewandelt, aber nicht ausgeführt. Doch für diesen Fall gibt es eine spezielle Taste ([Cursor-links]).
Zwei Sondertasten müssen Sie beim Umgang mit der Suchfunktion kennen:
[Cursor-links] löscht ein Zeichen
[End] beendet die Zeicheneingabe
Diese ungewöhnliche Tastenbelegung hat den Vorteil, den Code der Enter-Taste oder der Backspace-Taste mit nur einem Tastendruck eingeben zu können. Um andere Sonderzeichen einzufügen, hilft unter MS-DOS ein kleiner Trick. Halten Sie einfach die ALT-Taste gedrückt, geben den ASCII-Wert mit der Zehner-Tastatur ein und lassen die ALT-Taste wieder los. Diesen Code lesen Sie in der Hilfs-Tabelle nach (siehe [CTRL-F1]). Nachdem Sie nun die Zeichenfolge »abc« mit der End-Taste abgeschlossen haben, fragt Sie das Programm

```
In Großbuchstaben (j/n)
```

Wenn Sie nicht sicher sind, ob die Zeichenfolge, die Sie suchen, groß oder klein geschrieben ist, geben Sie [j] für »Ja« ein. Auf diese Weise werden alle Zeichen in der Datei vor dem Vergleich in Großbuchstaben umgewandelt. Eines ist dabei jedoch zu beachten: Umlaute werden nicht berücksichtigt. Sie können auch hier durch nochmaliges Drücken der Taste [End] den alten Wert übernehmen. Da in Ihrer Datei mit hoher Wahrscheinlichkeit keine Zeichenfolge »abc« steht, reagiert der Editor mit der Meldung »nichts gefunden«.
Haben Sie während des Suchvorgangs bemerkt, daß sich der Strich in der obersten Zeile des Bildes von »A« nach »E« bewegt hat? Das informiert Sie darüber, wo sich die Suchfunktion von Filed gerade befindet. Sie haben jederzeit die Möglichkeit, den Suchvorgang durch einen Tastendruck zu unterbrechen. Im linken Teil der letzten Bildschirmzeile wird diese Funktion während der Suche, angezeigt.

Wenn Sie bei der Eingabe der Zeichenfolge feststellen, daß der Suchvorgang doch nicht notwendig ist, können Sie mit [Esc] abbrechen, und die bis dahin eingegebenen Zeichen werden gelöscht. Findet das Programm die betreffende Zeichenfolge, so bleibt der Cursor an der betreffenden Position stehen.

Funktionstaste 4 [F4]: (Datei überschreiben, Ein- und Ausschalten)
Durch die Taste [F4] läßt sich der oben erwähnte Schreibschutz der Datei ein- beziehungsweise ausschalten. Betätigen Sie diese Taste zum erstenmal, so können Sie die aktuelle Datei überschreiben. Jeder weitere Tastendruck wird damit unwiderruflich in die Datei geschrieben. Das Zeichen, auf dem sich der Cursor zu diesem Zeitpunkt befindet, wird dabei überschrieben. In der untersten Bildschirmzeile wird dabei die Meldung »Schreiben = Ein« ausgegeben.

Überschreiben Sie nun probeweise einige Zeichen Ihrer – natürlich vorher kopierten – Datei. Schalten Sie die Funktion anschließend mit [F4] wieder aus und beobachten Sie, was passiert, wenn Sie nun erneut Zeichen eingeben. Sie stellen fest, daß der Computer so tut, als hätte niemand eine Taste gedrückt.

Funktionstaste 5 [F5]:
(Erneutes Suchen nach Zeichenfolge von [F3])
Mit [F5] wiederholen Sie durch einen einfachen Tastendruck die Suche nach dem Begriff, den Sie bereits eingegeben haben. Wurde noch keine Eingabe vorgenommen, so wird automatisch das Eingabefenster aktiviert. Erklärungen zur Eingabe lesen Sie unter [F3] nach. Auch hier läßt sich wieder durch einen Tastendruck die Suche unterbrechen.

Funktionstaste 6 [F6]:
(Aktive Seite im ASCII-Format anzeigen)
Mit [F6] werden die Daten, die Sie momentan auf dem Bildschirm sehen, entsprechend dem MS-DOS-Befehl »type« am Bildschirm aufgelistet. Dabei findet keine Umwandlung der Steuerzeichen statt. Auf diese Weise lassen sich auch reine ASCII-Dateien ausgeben. Wird bei der Darstellung im ASCII-Format der untere Bildschirmrand erreicht, so wartet der Computer auf einen Tastendruck, um anschließend eine weiter Seite anzuzeigen. Beendet wird die Funktion, wenn alle Zeichen des ursprünglichen Bildschirms bearbeitet worden sind. Dies kann sich ohne weiteres über drei bis vier Bildschirmseiten erstrecken.

Funktionstaste 7 [F7]:
(Finden und Ersetzen einer Zeichenfolge)
Wenn Sie während der Funktion [F7] den Modus »Schreiben = Aus« eingeschaltet haben, bekommen Sie eine Fehlermeldung. Das Programm kehrt nach einem Tastendruck zu dem Hauptschirm zurück. Mit einem weiteren Druck auf die F7-Taste wird die zu suchende Zeichenfolge – wie bei Funktion [F3] – eingeben. Darauf kommt ein zweites Fenster auf den Bildschirm, in dem Sie diejenige Zeichenfolge angeben, die die alte ersetzen soll. Gleichzeitig gibt das Programm eine Warnung aus, daß eine ungleiche Länge der beiden Zeichenfolgen ein unerwünschtes Überschreiben zur Folge haben kann. Die nötigen Information entnehmen Sie der Anzeige »Länge:«.

Schließen Sie nun die Eingabe mit [End] ab. Auf die Frage »Bis ans File-Ende ?« können Sie mit [j] oder [n] antworten. [n] ersetzt die erste gefundene Zeichenfolge ab der Cursor-Position, [j] hingegen wiederholt den Vorgang ohne Rückfrage bis zum Dateiende. [Esc] bringt Sie bei der

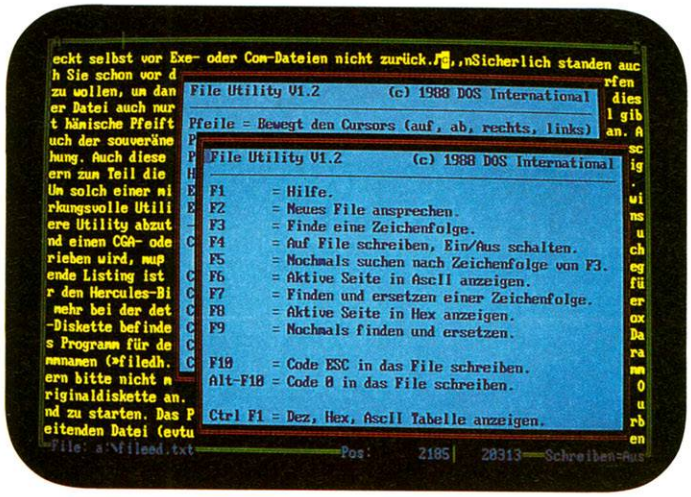


Bild 3. Funktionstastenbelegung von »Filed«

Eingabe von »suchen und ersetzen« zurück ins Hauptbild. Dabei werden die alten Such- und Ersetzungs-Zeichenfolgen gelöscht. Der Ersetzungs-Vorgang kann auch hier jederzeit mit einer Taste unterbrochen werden.

Beachten Sie den Strich auf der obersten Bildschirm-Zeile, der Ihnen wieder laufend die aktuelle Position der Suchfunktion angibt. In der untersten Zeile ändert sich die Angabe »Pos:« nur dann, wenn eine Zeichenfolge gefunden und ersetzt wurde. Auf diese Weise steht Ihnen eine zweite Kontrolle zur Verfügung.

Ist das Dateiende erreicht, wird entweder die Meldung »nichts gefunden« oder aber »... Zeichenfolgen ersetzt« ausgegeben. Der Cursor steht nun an der Position der zuletzt ersetzen Zeichenfolge. Bei einer erfolglosen Suche steht der Cursor an der Ausgangsposition der Suchfunktion.

Funktionstaste 8 [F8]:
(Aktive Seite in hex anzeigen)
Ab der Zeile, in der sich der Cursor befindet, wird Zeile um Zeile der Bildschirminhalt in hexadezimaler Schreibweise angezeigt. Darunter stehen jeweils die zugehörigen Originalzeichen:

```
57667662 <- High-Byte Hex
02F721D0 <- Low-Byte Hex
-------------------------
Program <- Text (ASCII)
```

In diesem Beispiel sehen Sie, daß der Buchstabe »P« den hexadezimalen Wert »h50« besitzt. In verschiedenen Programmiersprachen werden numerische Daten so gespeichert (gepackt), daß sie nur über solche Tabellen einzusehen sind.

Funktionstaste 9 [F9]:
(Erneut Suchen und Ersetzen der Zeichenfolge von [F7])
Hiermit läßt sich die Ersetzfunktion nach derselben Zeichenfolge, die bei [F7] eingegeben wurde, wiederholen. Haben Sie mit der Funktion [F7] noch keine Eingaben gemacht, so erhalten Sie automatisch wieder die Eingabefenster für die beiden Zeichenketten. Die Erklärungen zur Eingabe lesen Sie bitte unter [F3]. Auch hier kann durch einen Tastendruck die Ersetzfunktion abgebrochen werden.

Funktionstaste 10 [F10]:
(Esc-Code in die Datei schreiben)
Vielleicht haben Sie sich bereits gefragt, wie man es denn anstellt, einen Esc-Code in die Datei zu schreiben, wenn doch die Esc-Taste das Programm abschließt. Mit der Taste [F10] läßt sich dieser Code jederzeit eingefügen.

Alt-Funktionstaste 10 [Alt-F10]:
(Code 0 in die Datei schreiben)
Der hexadezimale Code Null benötigt ebenfalls eine eigene Taste, da dieser Code nicht über die Tastatur erreicht werden kann. Die Kombination aus der Alt- und der Null-Taste der Zehnertastatur hat keine Wirkung. Bei Filed läßt sich dieser Code bei gleichzeitig gedrückter Alt-Taste und [F10] auslösen. Erschrecken Sie aber nicht, wenn die Eingabe wie ein Leerzeichen aussieht. Mit [F8] können Sie den Wert Code »0« in hexadezimaler Darstellung betrachten. Der Null-Wert gibt aber kein spezielles Zeichen am Bildschirm aus.

Ctrl-Funktionstaste 1 [Ctrl-F1]:
(Dez-, Hex-, ASCII-Tabelle anzeigen)
Nach Drücken der beiden Tasten [Ctrl] und [F1] wird eine Tabelle mit dezimaler, hexadezimaler und ASCII-Darstellung aller 255 ASCII-Zeichen. Eine beliebige Taste blättert in dieser Tabelle weiter. Haben Sie das gesuchte Zeichen gefunden, so können Sie die Tabelle mit der Esc-Taste sofort verlassen.

Filed kann durch wenige Änderungen an einen PC mit Hercules-Monitor angepaßt werden

Das Programm ist, wie schon erwähnt, auf die CGA-Farbgrafikkarte voreingestellt. Besitzer von Hercules-Karten können aber das Programm an Ihren Computer anpassen. Hierfür müssen Sie in den Listings die in Klammern stehenden Werte eingeben. Ändern Sie vor allem die Konstante »GrafikKarte« von $B800 auf $B000.

Dazu kommen dann noch die Farbeinstellungen, die bei der Herculeskarte einen »unbrauchbaren« Wert haben. Am besten, Sie ändern diese ebenfalls in die Werte, die in Klammern stehen. Anwendern, denen die Einstellung der Funktionstasten nicht zusagt, steht es frei, diese abzuändern (Siehe Tastatur-Codes im Listing).

Um einen schnellen Zugriff zu erreichen, wird die angesprochene Datei mit zwei Variablen benutzt. Die eine besitzt 78 Zeichen Zeilenlänge. Damit wird mit einem einzigen Datei-Zugriff eine ganze Zeile verarbeitet.

Die andere Datei-Variable wurde mit nur einem Zeichen definiert. Damit finden Sie zum einen sofort die Dateilänge heraus, und zum anderen kann ein Zeichen direkt in die Datei geschrieben werden.

Einige Prozeduren müssen zum besseren Verständnis genauer erläutert werden.

Prozedur BufferLoeschen:
Bei schnellem, wiederholtem Blättern oder Cursorbewegen werden bei langsamen PCs ungewollt Reste aus dem Tastaturpuffer in die Datei geschrieben. Dies zerstört natürlich eine Datei. Um den fatalen Tatbestand zu umgehen, wird der Puffer einfach gelöscht. Die Reste des Puffers werden vor dem Verlassen der Prozedur zur Cursor-Bewegung einfach entfernt.

Prozedur Warte:
Nach Funktionsaufrufen wird mit dieser kleinen Prozedur solange gewartet, bis keine Taste mehr gedrückt wird. Tritt dies ein, wird mit derselben Funktionstaste weiter gearbeitet. Wenn Sie zum Beispiel eine ASCII-Anzeige der Datei wünschen ([F6]), so fahren Sie nach Anhalten der Ausgabe durch nochmaliges Drücken von [F6] mit dieser Funktion fort. »Warte« leert den Tastaturpuffer solange, bis keine Taste mehr gedrückt wird. Mit einer Funktionstaste werden ja zwei Tasten hintereinander in den Puffer geschrieben.

Prozedur Pos:
Errechnet die Position des Cursors innerhalb der Datei.

Prozedur SetCursor:

Unter bestimmten, großflächigen Zeichen in der Datei läßt sich der kleine Cursor nur unter großen Schwierigkeiten erkennen. »SetCursor« verändert den Cursor in ein großes, blinkendes Feld, das sofort ins Auge fällt.

Prozedur WoBinIch:

Gibt die Cursorposition in der obersten Bildschirmzeile an.

Prozedur GanzesBildAus:

Hiermit wird die Prozedur »RecordAusgabe« aufgerufen, die den gesamten Bildschirm mit den »File-Records« füllt. Damit aber bei nochmaligem Tastendruck (zum Beispiel Blättern) nur ein Teil des Bildschirmes gefüllt werden kann, kontrolliert die Prozedur auf »KeyPressed« hin. Wird ein Tastendruck bemerkt, so übergeht das Programm den Rest des Bildes und beginnt erneut oben mit dem Bildaufbau.

Prozedur Finde:

Es gibt sicher schnellere Suchroutinen. Doch wenn ein Wort (über eine Zeile hinaus) abgeschnitten wird, gestaltet dies die Suche bereits schwieriger. Die Lösung mit der Prozedur »Finde« ist dabei sicher am einfachsten und trotzdem genügend schnell.

Die übrigen Prozeduren sind leicht zu verstehen. Um Änderungen leicht zu ermöglichen, sind auch kleinste Befehle in einer separaten Prozedur untergebracht.

(Martin Baltensperger/sp)

```
[GL6]    0: Program FileUtility;
[000]    1:
[A71]    2: {$V-}
[351]    3: {$A+}
[H51]    4: {$C-}
[000]    5:
[M7G]    6: Const  GrafikKarte        = $B800; { $B800
           = Color / $B000 = Hercules }
[000]    7:
[0H5]    8:        ZeilenLaenge       = 78;
[CD5]    9:        ZeilenAnzahl       = 23;
[000]   10:
[416]   11:        { Farben ------------------ }
[000]   12:
[LRC]   13:        Rahmen             = 2;     {   2 =
           Color /  10 = Hercules }
[39D]   14:        HilfRahmen         = 4;     {   4 =
           Color /  10 = Hercules }
[D8D]   15:        HilfText           = 113;   { 113 =
           Color /  10 = Hercules }
[61F]   16:        HilfHinterGrund    = 7;     {   7 =
           Color /   2 = Hercules }
[N5C]   17:        Daten              = 14;    {  14 =
           Color /   2 = Hercules }
[8WB]   18:        Info               = 3;     {   3 =
           Color /   2 = Hercules }
[8BD]   19:        HexText            = 7;     {   7 =
           Color /  10 = Hercules }
[85F]   20:        FehlerHintergrund  = 1;     {   1 =
           Color /   1 = Hercules }
[0BD]   21:        FehlerText         = 28;    {  28 =
           Color /  10 = Hercules }
[2PD]   22:        EingabeFeld        = 113;   { 113 =
           Color / 120 = Hercules }
[000]   23:
[9F7]   24:        { Tastatur-Codes ---------- }
[000]   25:
[3Z3]   26:        Hilfe              = #59;
[DV4]   27:        NeuesFile          = #60;
[UD4]   28:        Finden             = #61;
[4T6]   29:        SchreibenEinAus    = #62;
[AC6]   30:        NochmalsFinden     = #63;
[AG5]   31:        AscIIAnzeigen      = #64;
[B96]   32:        FindenErsetzen     = #65;
[TA5]   33:        HexAnzeigen        = #66;
[286]   34:        NochmalsFindEr     = #67;
[105]   35:        EscEingabe         = #68;
[R95]   36:        NullEingabe        = #113;
[7X4]   37:        Tabelle            = #94;
[000]   38:
[NY4]   39:        LinksOben          = #119;
[A45]   40:        RechtsOben         = #132;
[Y65]   41:        LinksUnten         = #117;
[1D5]   42:        RechtsUnten        = #118;
[A75]   43:        ZeileAnfang        = #115;
[PW4]   44:        ZeileEnde          = #116;
[000]   45:
[HM3]   46:        Hoch               = #72;
[N24]   47:        Links              = #75;
[XG4]   48:        Rechts             = #77;
[4K4]   49:        Runter             = #80;
[XB5]   50:        SeiteRunter        = #81;
[4W4]   51:        SeiteHoch          = #73;
[5D4]   52:        Anfang             = #71;
[3M3]   53:        Ende               = #79;
[L25]   54:        SpezEinEnd         = #79;
```

```
[KN6]   55:        SpezEinZeichenDel  = #75;
[000]   56:
[EUA]   57:        HexBez             :  Array [0..15]
           of String [1] =
[Z56]   58:                             ('0','1','2','
           3','4','5','6','7','8','9',
[NZ3]   59:                              'A','B','C','
           D','E','F');
[000]   60:
[ETE]   61: Type   FileRecord         = Array [1..Zeile
           nLaenge] of Char;
[HL5]   62:        EinzelRec          = Char;
[SL7]   63:        AnyString          = String [80];
[1T4]   64:        Color              = Char;
[000]   65:
[129]   66: Var    FileName           : String [30];
[G8A]   67:        FileEin            : File of FileRec
           ord;
[DA8]   68:        FileBuffer         : FileRecord;
[EZA]   69:        FileEinEinzel      : File of EinzelR
           ec;
[000]   70:
[W25]   71:        Befehl             : Char;
[BG5]   72:        Eingabe            : Char;
[000]   73:
[8K5]   74:        Loop               : Integer;
[XM5]   75:        Loop1              : Integer;
[TV5]   76:        Zeile              : Integer;
[276]   77:        XPos, YPos         : Integer;
[KZ6]   78:        PosAnzeige         : Integer;
[MH5]   79:        ordi               : Integer;
[FH5]   80:        Hexi               : Integer;
[9T7]   81:        PosAnzeigeAlt      : Integer;
[RC5]   82:        SegM               : Integer;
[A16]   83:        OfSegM             : Integer;
[000]   84:
[8S6]   85:        PosCursorFile      : Real;
[OP5]   86:        PosRecord          : Real;
[SB6]   87:        PosObenLinks       : Real;
[RK6]   88:        FileRecAnzahl      : Real;
[ZM5]   89:        FileLaenge         : Real;
[FH5]   90:        WorkReal           : Real;
[000]   91:
[3F7]   92:        WasSuchen          : String [30];
[DL9]   93:        WasSuchenUmwandeln : String [30];
[7V7]   94:        WasErsetzen        : String [30];
[FE7]   95:        String8            : String [8];
[EC7]   96:        String3            : String [3];
[1C7]   97:        String2            : String [2];
[HWA]   98:        String78           : String [ZeilenL
           aenge];
[000]   99:
[DR6]  100:        Gefunden           : Boolean;
[FP8]  101:        GrossBuchstaben    : Boolean;
[VT6]  102:        GanzesFile         : Boolean;
[TS6]  103:        Schreiben          : Boolean;
[Z08]  104:        EinmalGefunden     : Boolean;
[000]  105:
[D4B]  106:        BildSpeicher       : Array [1..4000]
            of Byte;
[000]  107:
[GEC]  108: { ---------------------------------------
           -------------------------------- }
[000]  109:
[8LG]  110: Procedure FastDisp (S:AnyString; K,Z : Inte
           ger; Co : Color);
```

```
[000]  111:
[LND]  112: Type   BildSchirm          = Array [1..25,1.
           .80,1..2] of Char;
[000]  113:
[EHG]  114: Var    Display             : Bildschirm Abso
           lute GrafikKarte:0000;
[G05]  115:        I,J                 : Integer;
[000]  116:
[BH1]  117: begin
[RR3]  118:   J := length (S);
[2Y3]  119:   for I := 1 to J do
[BH1]  120:   begin
[376]  121:     Display [Z,K+I-1,2] := Co;
[CR6]  122:     Display [Z,K+I-1,1] := S[I];
[TU0]  123:   end;
[TU0]  124: end;
[000]  125:
[GEC]  126: { ---------------------------------------
           -------------------------------- }
[000]  127:
[RB8]  128: Procedure BufferLoeschen;
[000]  129:
[BG1]  130: Begin
[GW7]  131:   mem [0000:$041C] := mem [0000:$041A];
[UW7]  132:   mem [0000:$041D] := mem [0000:$041B];
[TU0]  133: end;
[000]  134:
[GEC]  135: { ---------------------------------------
           -------------------------------- }
[000]  136:
[ML5]  137: Procedure Warte;
[000]  138:
[BG1]  139: Begin
[4P5]  140:   While KeyPressed do
[U05]  141:   Read (kbd,Eingabe);
[TU0]  142: end;
[000]  143:
[GEC]  144: { ---------------------------------------
           -------------------------------- }
[000]  145:
[3M4]  146: Procedure Pos;
[000]  147:
[BG1]  148: Begin
[WFF]  149:   PosCursorFile := ((PosObenLinks-2)*Zeilen
           Laenge) +
[767]  150:                     ((YPos-1)*ZeilenLaenge)
[682]  151:                     + XPos-1;
[TU0]  152: end;
[000]  153:
[GEC]  154: { ---------------------------------------
           -------------------------------- }
[000]  155:
[4MV]  156: Procedure WindowOn (x,y,xx,yy,RahmenFarbe,W
           indowHintergrund:Integer; Sichern : Boolean)
           ;
[000]  157:
[TH6]  158: Var    Loop                : Integer;
[000]  159:
[BG1]  160: Begin
[Y65]  161:   If Sichern then
[EHD]  162:    Move (Mem[GrafikKarte:0000],Mem[SegM:0fS
           egM],4000);
[000]  163:
[7KA]  164:   FastDisp ('╔',x,y,chr(RahmenFarbe));
[EVA]  165:   FastDisp ('╚',x,yy,chr(RahmenFarbe));
[DRA]  166:   FastDisp ('╗',xx,y,chr(RahmenFarbe));
[DLB]  167:   FastDisp ('╝',xx,yy,chr(RahmenFarbe));
[6H6]  168:   For Loop := x+1 to xx-1 do
[BG1]  169:   Begin
[4UB]  170:     FastDisp ('═',Loop,y,chr(RahmenFarbe));
[KVB]  171:     FastDisp ('═',Loop,yy,chr(RahmenFarbe))
           ;
[TU0]  172:   end;
[XN6]  173:   For Loop := y+1 to yy -1 do
[BG1]  174:   Begin
[C5C]  175:     FastDisp ('║',x,Loop,chr(RahmenFarbe));
[32C]  176:     FastDisp ('║',xx,Loop,chr(RahmenFarbe))
           ;
[TU0]  177:   end;
[7W6]  178:   Window (x+1,y+1,xx-1,yy-1);
[B6B]  179:   TextBackGround (WindowHintergrund);
[NF2]  180:   ClrScr;
[V85]  181:   TextColor (Black);
[TU0]  182: end;
[000]  183:
[GEC]  184: { ---------------------------------------
           -------------------------------- }
[000]  185:
[ZC6]  186: Procedure WindowOff;
[000]  187:
[BG1]  188: Begin
[V43]  189:   Normvideo;
[S57]  190:   TextBackGround (Black);
[824]  191:   Window (1,1,80,25);
[NF2]  192:   ClrScr;
[2DD]  193:   Move (Mem[SegM:0fSegM],Mem[GrafikKarte:00
           00],4000);
[TU0]  194: end;
[000]  195:
[GEC]  196: { ---------------------------------------
           -------------------------------- }
[000]  197:
[NEC]  198: Procedure SetCursor (top,bottom:byte);
[000]  199:
[G27]  200: var    vport               : integer;
[000]  201:
[BH1]  202: begin
[2G6]  203:   Vport := MemW [$0040:$0063] ;
[9D4]  204:   Port [Vport]:= 10;
[OM5]  205:   Port [Vport+1]:= Top;
[AD4]  206:   Port [Vport]:= 11;
[Y87]  207:   Port [Vport+1]:= Bottom
[TU0]  208: end;
[000]  209:
[GEC]  210: { ---------------------------------------
           -------------------------------- }
[000]  211:
[XW5]  212: Procedure WoBinIch;
[000]  213:
[BG1]  214: Begin
[LV0]  215:   Pos;
[9P8]  216:   Str (PosCursorFile:8:0,String8);
[EN9]  217:   FastDisp (String8,47,25,chr(Info));
[CLE]  218:   PosAnzeige := Trunc ((PosCursorFile/FileL
           aenge)*73);
[L1A]  219:   IF PosAnzeige <> PosAnzeigeAlt then
[BG1]  220:   Begin
[7LC]  221:     FastDisp ('=',PosAnzeigeAlt+4,1,Chr(Rah
           men));
[RCC]  222:     FastDisp ('╪',PosAnzeige+4,1,Chr(Rahmen
           ));
[8F8]  223:     PosAnzeigeAlt := PosAnzeige;
[TU0]  224:   end;
[TU0]  225: end;
[000]  226:
[GEC]  227: { ---------------------------------------
           -------------------------------- }
[000]  228:
[X3B]  229: Procedure BildSchirm (Woher:Integer);
[000]  230:
[BG1]  231: Begin
[YKY]  232:   FastDisp ('╔A═══════════════════════════
           ═══════════════════════════════════
           ═══E╗',1,1,Chr(Rahmen));
[7M4]  233:   If Woher = 0 then
[SR5]  234:    For Loop := 2 to 24 do
[YSK]  235:    FastDisp ('║
                  ',1,Loop,Chr(Rahmen));
[VVZ]  236:   FastDisp ('╚═══════════════════════════
           ═══════════════════════════════════
           ═══╝',1,25,Chr(Rahmen));
[E59]  237:   FastDisp ('File: ',3,25,Chr(Info));
[WD9]  238:   FastDisp (FileName,9,25,Chr(Info));
[F19]  239:   FastDisp ('Pos:  ',41,25,Chr(Info));
[L38]  240:   Str (FileLaenge:8:0,String8);
[DN9]  241:   FastDisp (String8,56,25,chr(Info));
[G75]  242:   If Schreiben then
[DXD]  243:    FastDisp ('Schreiben=Ein',67,25,Chr(Fehl
           erText))
[A11]  244:   else
[4UB]  245:    FastDisp ('Schreiben=Aus',67,25,Chr(Info
           ));
[7C5]  246:   PosAnzeigeAlt := -1;
[TU0]  247: end;
[000]  248:
[GEC]  249: { ---------------------------------------
           -------------------------------- }
[000]  250:
[LYA]  251: Procedure FileOn (Woher:Integer);
[000]  252:
[BG1]  253: Begin
[4W6]  254:   If Woher <> 1 then ClrScr;
[E6B]  255:   If (ParamCount = 0) or (Woher = 1) then
[BG1]  256:   Begin
[WOF]  257:     WindowOn (30,1,80,5,HilfRahmen,HilfHint
           erGrund,True);
[PY2]  258:     GotoXY (2,2);
[PHG]  259:     Write ('Welches File: _________________
           _____');
[SZ2]  260:     BufLen := 30;
[383]  261:     GotoXY (16,2);
[N55]  262:     Readln (FileName);
[XZ2]  263:     WindowOff;
```

```
[DLO]  264:    end
[A11]  265:    else
[FP6]  266:     FileName := ParamStr (1);
[000]  267:
[KX7]  268:    Assign (FileEin,FileName);
[1K6]  269:    {$I-} Reset (FileEin); {$I+}
[PG5]  270:    If IoResult <> 0 then
[BG1]  271:    Begin
[NF2]  272:      ClrScr;
[WK9]  273:      Writeln ('File nicht gefunden!');
[5B1]  274:      halt;
[TU0]  275:    end;
[LR9]  276:    Assign (FileEinEinzel,FileName);
[X46]  277:    Reset (FileEinEinzel);
[TU0]  278: end;
[000]  279:
[GEC]  280: { ---------------------------------------
       ---------------------------------- }
[000]  281:
[EUB]  282: Procedure RecordAusgabe (Y : Integer);
[000]  283:
[BH1]  284: begin
[PW6]  285:    If not Eof (FileEin) then
[BG1]  286:    Begin
[YW6]  287:      Read (FileEin,FileBuffer);
[EB9]  288:      For Loop := 1 to ZeilenLaenge do
[CAB]  289:       String78 [Loop] := FileBuffer [Loop];
[X2A]  290:      FastDisp (String78,2,Y+1,Chr(Daten));
[TU0]  291:    end;
[TU0]  292: end;
[000]  293:
[GEC]  294: { ---------------------------------------
       ---------------------------------- }
[000]  295:
[XXB]  296: Procedure GanzesBildAus (Woher:Byte);
[000]  297:
[BG1]  298: Begin
[HC2]  299:    Zeile := 1;
[WY7]  300:    Seek (FileEin,PosObenLinks-1);
[X42]  301:    Repeat
[XOC]  302:      If (not KeyPressed) or (Woher = 0) then
[N26]  303:       RecordAusgabe (Zeile);
[ZC5]  304:      Zeile := Succ(Zeile);
[BLC]  305:    until (Zeile > ZeilenAnzahl) or (Eof (Fil
       eEin));
[TU0]  306: end;
[000]  307:
[GEC]  308: { ---------------------------------------
       ---------------------------------- }
[000]  309:
[6N9]  310: Procedure AscIIAus (Y : Integer);
[000]  311:
[BH1]  312: begin
[BK8]  313:    Seek (FileEin,PosObenLinks+(Y-2));
[PW6]  314:    If not Eof (FileEin) then
[BG1]  315:    Begin
[YW6]  316:      Read (FileEin,FileBuffer);
[EB9]  317:      For Loop := 1 to ZeilenLaenge do
[BG1]  318:      Begin
[T75]  319:       If KeyPressed then
[721]  320:        EXIT;
[000]  321:
[EXD]  322:       If (FileBuffer [Loop] = Chr(10)) or (W
       hereX = 78) then
[A45]  323:        Loop1 := Succ (Loop1);
[NY8]  324:       If Loop1 = ZeilenAnzahl then
[BG1]  325:       Begin
[U05]  326:        Read (kbd,Eingabe);
[L52]  327:        Loop1 := 0;
[UW1]  328:        Warte;
[TU0]  329:       end;
[DL7]  330:       Write (FileBuffer [Loop]);
[TU0]  331:      end;
[TU0]  332:    end;
[TU0]  333: end;
[000]  334:
[GEC]  335: { ---------------------------------------
       ---------------------------------- }
[000]  336:
[WM7]  337: Procedure AscIIAnzeige;
[000]  338:
[BG1]  339: Begin
[W34]  340:    Window (2,2,79,24);
[NF2]  341:    ClrScr;
[DWN]  342:    FastDisp ('Taste unterbricht / Bei Halt -
       > Taste drücken',2,25,Chr(Info));
[L52]  343:    Loop1 := 0;
[HC2]  344:    Zeile := 1;
[X42]  345:    Repeat
[Z15]  346:      AscIIAus (Zeile);
[ZC5]  347:      Zeile := Succ(Zeile);
```

```
[LDH]  348:    until (Zeile > ZeilenAnzahl) or (Eof (Fil
       eEin)) or (KeyPressed);
[U05]  349:    Read (kbd,Eingabe);
[NF2]  350:    ClrScr;
[824]  351:    Window (1,1,80,25);
[8V3]  352:    BildSchirm(1);
[TT4]  353:    GanzesBildAus(0);
[TU0]  354: end;
[000]  355:
[GEC]  356: { ---------------------------------------
       ---------------------------------- }
[000]  357:
[9N6]  358: Procedure HexAnzeige;
[000]  359:
[GH6]  360: Var    Work               : Integer;
[000]  361:
[BG1]  362: Begin
[W34]  363:    Window (2,2,79,24);
[NF2]  364:    ClrScr;
[07W]  365:    FastDisp ('-----------------------------
       ----------------------------------------
       ----',2,23,Chr(Rahmen));
[BVP]  366:    FastDisp (
       ',2,24,Chr(Rahmen));
[UW3]  367:    Work := YPos-2;
[3Y9]  368:    Seek (FileEin,PosObenLinks+(Work-1));
[BA5]  369:    for Loop1 := 0 to 6 do
[A27]  370:    If Not Eof(FileEin) then
[BH1]  371:    begin
[YW6]  372:      Read (FileEin,FileBuffer);
[UD9]  373:      For Work := 1 to ZeilenLaenge do
[NAB]  374:       String78 [Work] := FileBuffer [Work];
[00B]  375:      For Loop := 1 to Length (String78) do
[BG1]  376:      Begin
[UL7]  377:        Ordi := ord (String78 [Loop]);
[416]  378:        Hexi := Trunc(ordi/16);
[VEF]  379:        FastDisp (HexBez [Hexi],Loop+1,Loop1*
       3+2,Chr(HexText));
[NDK]  380:        FastDisp (HexBez [ordi-(hexi*16)],Loo
       p+1,Loop1*3+3,Chr(HexText));
[TU0]  381:      end;
[NO4]  382:      Loop := Loop1*3+4;
[XTA]  383:      FastDisp (String78,2,Loop,Chr(Daten));
[TU0]  384:    end;
[KY2]  385:    GotoXY (1,1);
[U05]  386:    Read (kbd,Eingabe);
[NF2]  387:    ClrScr;
[724]  388:    Window (1,1,80,24);
[TT4]  389:    GanzesBildAus(0);
[VZ2]  390:    NormVideo;
[TU0]  391: end;
[000]  392:
[GEC]  393: { ---------------------------------------
       ---------------------------------- }
[000]  394:
[07T]  395: Function SpezEin (x,y,len:Integer; co:Color
       ;AlterString:AnyString): AnyString;
[000]  396:
[GH6]  397: Var    Work               : Integer;
[XM5]  398:        Work1              : Integer;
[T25]  399:        xx                 : Integer;
[U98]  400:        WorkString         : AnyString;
[AC8]  401:        FunktionsTaste     : Boolean;
[000]  402:
[BG1]  403: Begin
[HN6]  404:    For Work := 1 to Len do
[WNB]  405:     FastDisp ('_',x+Work-1,y,Chr(EingabeFeld
       ));
[ML7]  406:    WorkString := AlterString;
[Z88]  407:    xx := x+Length (AlterString);
[NU9]  408:    FastDisp ('Länge:',5,y-1,Chr(Info));
[C5D]  409:    Fastdisp (AlterString,X,Y,Chr(EingabeFeld
       ));
[X42]  410:    Repeat
[BE5]  411:      Str (XX-14:2,String2);
[LV9]  412:      FastDisp (String2,11,Y-1,Chr(Info));
[AG3]  413:      GotoXY (XX,Y);
[UT4]  414:      Read (Kbd,Eingabe);
[T18]  415:      FunktionsTaste := false;
[64B]  416:      If (Eingabe = #27) and not Keypressed t
       hen
[721]  417:       EXIT;
[000]  418:
[RPA]  419:      If (Eingabe = #27) and Keypressed then
[BG1]  420:      Begin
[UT4]  421:        Read (Kbd,Eingabe);
[UR7]  422:        FunktionsTaste := True;
[W3A]  423:        If Eingabe = SpezEinZeichenDel then
[BG1]  424:        Begin
[YD7]  425:          If (XX-1 < X+Len) and (XX > X) then
```

```
[HYA]  426:          FastDisp ('_',XX-1,y,Chr(EingabeFel
       d));
[OT3]  427:          If XX > X then
[BG1]  428:          Begin
[TMG]  429:            WorkString := Copy (WorkString,1,
       Length(WorkString)-1);
[1E3]  430:            XX := Pred(XX);
[TUO]  431:          end;
[DLO]  432:        end
[A11]  433:        else
[WR8]  434:        If Eingabe = NullEingabe then
[BG1]  435:        Begin
[WV4]  436:         Eingabe := Chr(0);
[TY7]  437:         FunktionsTaste := False;
[DLO]  438:        end
[A11]  439:        else
[U78]  440:        If Eingabe = EscEingabe then
[BG1]  441:        Begin
[RO5]  442:         Eingabe := Chr(27);
[TY7]  443:         FunktionsTaste := False;
[TUO]  444:        end;
[TUO]  445:      end;
[000]  446:
[038]  447:      If not FunktionsTaste then
[BG1]  448:      Begin
[AB4]  449:        If XX < X+Len then
[BG1]  450:        Begin
[F1C]  451:          FastDisp (Eingabe,XX,Y,Chr(EingabeF
       eld));
[YBA]  452:          WorkString := WorkString + Eingabe;
[KE3]  453:          XX := Succ(XX);
[TUO]  454:        end;
[8Y3]  455:        Eingabe := ' ';
[TUO]  456:      end;
[XH7]  457:    Until Eingabe = SpezEinEnd;
[Z77]  458:    SpezEin := WorkString;
[TTO]  459: End;
[000]  460:
[GEC]  461: { ----------------------------------------
       ---------------------------------- }
[000]  462:
[3Z5]  463: Procedure FindeEin;
[000]  464:
[GH6]  465: Var    Work                : Integer;
[000]  466:
[3Z8]  467: Procedure GrossBuchstabeJN;
[BG1]  468: Begin
[P78]  469:    If GrossBuchstaben then
[779]  470:     FastDisp ('J',68,5,Chr(EingabeFeld))
[A11]  471:    else
[YG9]  472:     FastDisp ('N',68,5,Chr(EingabeFeld));
[TUO]  473: end;
[000]  474:
[BG1]  475: Begin
[17A]  476:    WindowOn (3,3,78,6,HilfRahmen,0,True);
[824]  477:    Window (1,1,80,25);
[C89]  478:    FastDisp ('Suche:',5,5,Chr(Info));
[2BD]  479:    WasSuchen := SpezEin (14,5,30,Chr(Info),W
       asSuchen);
[5B5]  480:    if Eingabe = #27 then
[721]  481:     EXIT;
[000]  482:
[80F]  483:    FastDisp ('In GrossBuchstaben : _ (j/n)',
       47,5,Chr(Info));
[TF5]  484:    GrossBuchstabeJN;
[S83]  485:    GotoXY (68,5);
[U05]  486:    Read (kbd,Eingabe);
[1U4]  487:    If Eingabe = '' then
[8Y3]  488:     Eingabe := ' ';
[RLA]  489:    If (Eingabe = #27) and KeyPressed then
[BG1]  490:    Begin
[U05]  491:      Read (kbd,Eingabe);
[8Y3]  492:      Eingabe := ' ';
[TUO]  493:    end;
[SZ7]  494:    Eingabe := UpCase (Eingabe);
[5G5]  495:    If Eingabe = 'J' then
[T68]  496:     GrossBuchstaben := True;
[1H5]  497:    If Eingabe = 'N' then
[LB8]  498:     Grossbuchstaben := False;
[TF5]  499:    GrossBuchstabeJN;
[F7A]  500:    WasSuchenUmwandeln := WasSuchen;
[P78]  501:    If GrossBuchstaben then
[NYE]  502:     For Work := 1 to Length (WasSuchenUmwand
       eln) do
[UBM]  503:       WasSuchenUmwandeln [Work] := UpCase (Wa
       sSuchenUmwandeln [Work]);
[TUO]  504: end;
[000]  505:
[GEC]  506: { ----------------------------------------
       ---------------------------------- }
[000]  507:
```

```
[GEA]  508: Procedure Finde (Woher:integer);
[000]  509:
[3S8]  510: Var    NaechsteZeile       : Boolean;
[2N5]  511:        Loop2               : Integer;
[2S3]  512:        I                   : Real;
[8K5]  513:        Work                : Integer;
[LN5]  514:        RealWork            : Real;
[000]  515:
[BG1]  516: Begin
[43A]  517:   If (Woher = 0) or (WasSuchen = '') then
[BG1]  518:   Begin
[3N2]  519:    FindeEin;
[XZ2]  520:    WindowOff;
[5A5]  521:    If Eingabe = #27 then
[BG1]  522:    Begin
[W04]  523:     WasSuchen := '';
[721]  524:     EXIT;
[TUO]  525:    end;
[TUO]  526:   end;
[000]  527:
[BK4]  528:   If Woher <> 9 then
[XYK]  529:    FastDisp ('Suche begonnen! (Taste unterb
       richt)',2,25,Chr(Info))
[A11]  530:   else
[DR7]  531:   If not EinmalGefunden then
[OZK]  532:    FastDisp ('Ersetzen begonnen! (Taste unt
       erbricht)',2,25,Chr(Info));
[000]  533:
[N47]  534:   I := PosObenLinks + (YPos-3);
[344]  535:   Seek (FileEin,I);
[PA5]  536:   gefunden := false;
[542]  537:   Loop := 1;
[40B]  538:   Loop1 := Length(WasSuchenUmwandeln);
[8E7]  539:   NaechsteZeile := False;
[P83]  540:   Work := XPos;
[NZC]  541:   while (FileRecAnzahl > I) and (not gefund
       en) do
[BH1]  542:   begin
[2W4]  543:     If Loop > 1 then
[WW6]  544:      NaechsteZeile := true;
[YW6]  545:     Read (FileEin,FileBuffer);
[4Z9]  546:     If KeyPressed then Gefunden := true;
[000]  547:
[RBC]  548:     PosCursorFile := (I*ZeilenLaenge)+Work;
[CLE]  549:     PosAnzeige := Trunc ((PosCursorFile/Fil
       eLaenge)*73);
[L1A]  550:     IF PosAnzeige <> PosAnzeigeAlt then
[BG1]  551:     Begin
[7LC]  552:       FastDisp ('=',PosAnzeigeAlt+4,1,Chr(R
       ahmen));
[RCC]  553:       FastDisp ('╪',PosAnzeige+4,1,Chr(Rahm
       en));
[8F8]  554:       PosAnzeigeAlt := PosAnzeige;
[TUO]  555:     end;
[000]  556:
[MBA]  557:     While (Work < 79) and (not gefunden) do
[BG1]  558:     Begin
[OY6]  559:       If ((GrossBuchstaben) and
[VFH]  560:           (WasSuchenUmwandeln [Loop] = Upcas
       e(FileBuffer [Work]))) or
[Y57]  561:          ((not GrossBuchstaben) and
[WLG]  562:          (WasSuchenUmwandeln [Loop] = FileB
       uffer [Work])) then
[BG1]  563:       Begin
[436]  564:         If loop = Loop1 then
[BH1]  565:         begin
[1C5]  566:           Gefunden := true;
[3T3]  567:           Loop2 := work;
[DLO]  568:         end
[A11]  569:         else
[WP4]  570:          Loop := Succ(Loop);
[DLO]  571:       end
[A11]  572:       else
[542]  573:        Loop := 1;
[TS4]  574:       Work := Succ(Work);
[TUO]  575:     end;
[MM1]  576:     I := I + 1;
[552]  577:     work := 1;
[TUO]  578:   end;
[8G1]  579:   I := I -1;
[MZ3]  580:   RealWork := I;
[PU8]  581:   I := (I*ZeilenLaenge)+Loop2-loop1;
[PW6]  582:   If I >= FileLaenge then
[P45]  583:    Gefunden := False;
[000]  584:
[N2C]  585:   If (gefunden) and (not KeyPressed) then
[BH1]  586:   begin
[STD]  587:     If RealWork < PosObenLinks + ZeilenAnza
       hl-1  then
[BH1]  588:     begin
[9FA]  589:       YPos := Trunc(RealWork-PosObenLinks);
```

```
[18C]  590:        XPos := Trunc(I-(RealWork*ZeilenLaeng
           e))+2;
[KP3]  591:        YPos := YPos +3;
[DT4]  592:        If XPos < 2 then
[BG1]  593:        Begin
[684]  594:          XPos := 2-XPos;
[K38]  595:          XPos := ZeilenLaenge+2 - XPos;
[GP3]  596:          YPos := YPos -1;
[TU0]  597:        end;
[DL0]  598:      end
[A11]  599:      else
[BH1]  600:      begin
[VT7]  601:        PosObenLinks := Realwork;
[BK4]  602:        If Woher <> 9 then
[BG1]  603:        Begin
[W34]  604:          Window (2,2,79,24);
[NF2]  605:          ClrScr;
[NT4]  606:          GanzesBildAus (0);
[824]  607:          Window (1,1,80,25);
[TU0]  608:        end;
[18C]  609:        XPos := Trunc(I-(RealWork*ZeilenLaeng
           e))+2;
[232]  610:        YPos := 3;
[DT4]  611:        If XPos < 2 then
[BG1]  612:        Begin
[684]  613:          XPos := 2-XPos;
[K38]  614:          XPos := ZeilenLaenge+2 - XPos;
[GP3]  615:          YPos := YPos -1;
[TU0]  616:        end;
[TU0]  617:      end;
[DL0]  618:    end
[A11]  619:    else
[BK4]  620:    If Woher <> 9 then
[BG1]  621:    Begin
[W77]  622:      If not KeyPressed then
[BH1]  623:      begin
[5LG]  624:        WindowOn (3,18,39,22,HilfRahmen,Fehle
           rHinterGrund,True);
[Y2L]  625:        FastDisp ('Nichts gefunden ! (Taste d
           rücken)',5,20,Chr(FehlerText));
[U05]  626:        Read (kbd,Eingabe);
[XZ2]  627:        WindowOff;
[DL0]  628:      end
[A11]  629:      else
[BG1]  630:      Begin
[A0B]  631:        WindowOn (3,18,56,22,HilfRahmen,0,Tru
           e);
[T3R]  632:        FastDisp ('Zeichenfolge suchen unterb
           rochen ! (Taste drücken)',5,20,Chr(Info));
[UW1]  633:        Warte;
[U05]  634:        Read (kbd,Eingabe);
[XZ2]  635:        WindowOff;
[TU0]  636:      end;
[TU0]  637:    end;
[BK4]  638:    If Woher <> 9 then
[BG1]  639:    Begin
[544]  640:      BildSchirm (1);
[DL0]  641:    end
[A11]  642:    else
[T75]  643:     If KeyPressed then
[PA5]  644:      gefunden := false;
[TU0]  645: end;
[000]  646:
[GEC]  647: { ------------------------------------------
           -------------------------------- }
[000]  648:
[2A5]  649: Procedure Down;
[000]  650:
[BG1]  651: Begin
[299]  652:   PosObenLinks := PosObenLinks + 1;
[Z5G]  653:   If PosObenLinks > FileRecAnzahl - ZeilenA
           nzahl then
[BG1]  654:   Begin
[DNE]  655:    PosObenLinks := (FileRecAnzahl - ZeilenA
           nzahl) +1;
[NT4]  656:    GanzesBildAus (0);
[DL0]  657:   end
[A11]  658:   else
[BH1]  659:   begin
[W34]  660:     Window (2,2,79,24);
[583]  661:     GotoXY (1,23);
[9Z2]  662:     writeln;
[DU4]  663:     RecordAusgabe (23);
[724]  664:     Window (1,1,80,24);
[TU0]  665:   end;
[TU0]  666: end;
[000]  667:
[GEC]  668: { ------------------------------------------
           -------------------------------- }
[000]  669:
[NE8]  670: Procedure Up (Woher:Integer);
[000]  671:
```

```
[BG1]  672: Begin
[AB9]  673:   PosObenLinks := PosObenLinks -1;
[RK7]  674:   If PosObenLinks < 1 then
[DX4]  675:   PosObenLinks := 1;
[V76]  676:   GanzesBildAus (Woher);
[TT0]  677: End;
[000]  678:
[GEC]  679: { ------------------------------------------
           -------------------------------- }
[000]  680:
[8ZA]  681: Procedure PfeilAuf (Woher:Integer);
[000]  682:
[BG1]  683: Begin
[GK4]  684:   YPos := pred(YPos);
[HU4]  685:   if YPos < 2 then
[BH1]  686:   begin
[132]  687:     YPos := 2;
[FP2]  688:     Up (Woher);
[TU0]  689:   end;
[TU0]  690: end;
[000]  691:
[GEC]  692: { ------------------------------------------
           -------------------------------- }
[000]  693:
[TR5]  694: Procedure PfeilAb;
[000]  695:
[BG1]  696: Begin
[LV0]  697:   Pos;
[AGF]  698:   If PosCursorFile + ZeilenLaenge <= FileLa
           enge then
[FK4]  699:    YPos := succ(YPos);
[AP4]  700:   if YPos > 24 then
[BH1]  701:   begin
[482]  702:     YPos := 24;
[6B1]  703:     Down;
[TU0]  704:   end;
[TU0]  705: end;
[000]  706:
[GEC]  707: { ------------------------------------------
           -------------------------------- }
[000]  708:
[UW6]  709: Procedure PfeilLinks;
[000]  710:
[BG1]  711: Begin
[2E7]  712:   If PosCursorFile >1 then
[9K4]  713:    XPos := pred(XPos);
[DT4]  714:   If XPos < 2 then
[BG1]  715:   Begin
[K82]  716:     XPos := 79;
[FC3]  717:     PfeilAuf (0);
[TU0]  718:   end;
[TU0]  719: end;
[000]  720:
[GEC]  721: { ------------------------------------------
           -------------------------------- }
[000]  722:
[NF7]  723: Procedure PfeilRechts;
[000]  724:
[BG1]  725: Begin
[V9A]  726:   If PosCursorFile < FileLaenge then
[8K4]  727:    XPos := succ(XPos);
[9P4]  728:   If XPos > 79 then
[BG1]  729:   Begin
[032]  730:     XPos := 2;
[ZH2]  731:     PfeilAb;
[TU0]  732:   end;
[TU0]  733: end;
[000]  734:
[GEC]  735: { ------------------------------------------
           -------------------------------- }
[000]  736:
[725]  737: Procedure PgDn;
[000]  738:
[BG1]  739: Begin
[4SG]  740:   If (PosObenLinks-1) + ZeilenAnzahl < File
           RecAnzahl then
[BG1]  741:   Begin
[5SC]  742:     PosObenLinks := PosObenLinks + ZeilenAn
           zahl;
[78H]  743:     if PosObenLinks > (FileRecAnzahl - Zeil
           enAnzahl) then
[TTE]  744:     PosObenLinks := (FileRecAnzahl - Zeilen
           Anzahl) + 1;
[TU0]  745:   end;
[NT4]  746:   GanzesBildAus (1);
[TU0]  747: end;
[000]  748:
[GEC]  749: { ------------------------------------------
           -------------------------------- }
[000]  750:
[855]  751: Procedure PgUp;
[000]  752:
```

```
[BG1]  753: Begin
[HSC]  754:   PosObenLinks := PosObenLinks - ZeilenAnza
       hl;
[RK7]  755:   If PosObenLinks < 1 then
[DX4]  756:    PosObenLinks := 1;
[NT4]  757:   GanzesBildAus (1);
[TUO]  758: end;
[000]  759:
[GEC]  760: { ----------------------------------------
       ---------------------------------- }
[000]  761:
[GB5]  762: Procedure GoEnd;
[000]  763:
[BG1]  764: Begin
[00E]  765:   PosObenLinks := FileRecAnzahl - ZeilenAnz
       ahl+1;
[RK7]  766:   If PosObenLinks < 1 then
[DX4]  767:   PosObenLinks := 1;
[W34]  768:   Window (2,2,79,24);
[NF2]  769:   ClrScr;
[824]  770:   Window (1,1,80,25);
[NT4]  771:   GanzesBildAus (0);
[TUO]  772: end;
[000]  773:
[GEC]  774: { ----------------------------------------
       ---------------------------------- }
[000]  775:
[4H5]  776: Procedure GoBegin;
[000]  777:
[BG1]  778: Begin
[DX4]  779:   PosObenLinks := 1;
[NT4]  780:   GanzesBildAus (0);
[TUO]  781: end;
[000]  782:
[GEC]  783: { ----------------------------------------
       ---------------------------------- }
[000]  784:
[PK5]  785: Procedure ZAnfang;
[000]  786:
[BG1]  787: Begin
[052]  788:   Xpos := 2;
[TUO]  789: end;
[000]  790:
[GEC]  791: { ----------------------------------------
       ---------------------------------- }
[000]  792:
[TC5]  793: Procedure ZEnde;
[000]  794:
[BG1]  795: Begin
[BB6]  796:   XPos := ZeilenLaenge +1;
[X42]  797:   Repeat
[LVO]  798:     Pos;
[Z9A]  799:     If PosCursorFile > FileLaenge then
[9H4]  800:     XPos := Pred(XPos);
[YW9]  801:   until PosCursorFile <= FileLaenge;
[TUO]  802: end;
[000]  803:
[GEC]  804: { ----------------------------------------
       ---------------------------------- }
[000]  805:
[KD5]  806: Procedure ROben;
[000]  807:
[BG1]  808: Begin
[132]  809:   YPos := 2;
[TS1]  810:   ZEnde;
[TUO]  811: end;
[000]  812:
[GEC]  813: { ----------------------------------------
       ---------------------------------- }
[000]  814:
[OD5]  815: Procedure LOben;
[000]  816:
[BG1]  817: Begin
[052]  818:   Xpos := 2;
[132]  819:   YPos := 2;
[TUO]  820: end;
[000]  821:
[GEC]  822: { ----------------------------------------
       ---------------------------------- }
[000]  823:
[AK5]  824: Procedure LUnten;
[000]  825:
[BG1]  826: Begin
[052]  827:   Xpos := 2;
[UE6]  828:   YPos := ZeilenAnzahl+1;
[VXC]  829:   While PosObenLinks + YPos -2 > FileRecAnz
       ahl do
[GH4]  830:    YPos := Pred(YPos);
[TUO]  831: end;
[000]  832:
[GEC]  833: { ----------------------------------------
       ---------------------------------- }
```

```
[000]  834:
[WK5]  835: Procedure RUnten;
[000]  836:
[BG1]  837: Begin
[BB6]  838:   XPos := ZeilenLaenge +1;
[UE6]  839:   YPos := ZeilenAnzahl+1;
[VXC]  840:   While PosObenLinks + YPos -2 > FileRecAnz
       ahl do
[GH4]  841:    YPos := Pred(YPos);
[LVO]  842:   Pos;
[UVG]  843:   If (PosCursorFile > FileLaenge) and (File
       RecAnzahl > 1) then
[GH4]  844:   YPos := Pred(YPos);
[TS1]  845:   ZEnde;
[TUO]  846: end;
[000]  847:
[GEC]  848: { ----------------------------------------
       ---------------------------------- }
[000]  849:
[EKC]  850: Procedure FindeErsetze (Woher:Integer);
[000]  851:
[X57]  852: Var    Anzahl              : Integer;
[786]  853:        Laenge              : Integer;
[000]  854:
[BG1]  855: Begin
[3H6]  856:   If not Schreiben then
[BG1]  857:   Begin
[GVG]  858:     WindowOn (38,20,80,25,HilfRahmen,Fehler
       Hintergrund,True);
[NGM]  859:     FastDisp ('Fehler: In Modus (Finden/Ers
       etzen) muss',40,21,Chr(FehlerText));
[48M]  860:     FastDisp ('        der Schalter (Schrei
       ben Ein/Aus)',40,22,Chr(FehlerText));
[K7M]  861:     FastDisp ('        eingeschaltet sein,
       sonst kann',40,23,Chr(FehlerText));
[F8K]  862:     FastDisp ('        nicht ersetzt werden
       !',40,24,Chr(FehlerText));
[U05]  863:     Read (kbd,Eingabe);
[XZ2]  864:     WindowOff;
[7B1]  865:     Exit;
[TUO]  866:   end;
[W7A]  867:   If (Woher = 0) or (WasErsetzen = '') then
[BG1]  868:   Begin
[3N2]  869:     FindeEin;
[5A5]  870:     If Eingabe = #27 then
[BG1]  871:     Begin
[WO4]  872:      WasSuchen := '';
[GK4]  873:      WasErsetzen := '';
[XZ2]  874:      WindowOff;
[721]  875:      EXIT;
[TUO]  876:     end;
[000]  877:
[Y3G]  878:     WindowOn (42,13,78,23,HilfRahmen,HilfHi
       nterGrund,False);
[DWG]  879:     FastDisp ('Beachte: Wenn die Länge des'
       ,44,14,Chr(HilfText));
[U4K]  880:     FastDisp ('        SuchArgumentes ungl
       eich',44,15,Chr(HilfText));
[A1L]  881:     FastDisp ('        des ErsetzungsArgum
       entes',44,16,Chr(HilfText));
[CMH]  882:     FastDisp ('        ist, wird trotzdem
       kein',44,17,Chr(HilfText));
[7XE]  883:     FastDisp ('        Fehler Angezeigt.',
       44,18,Chr(HilfText));
[SXF]  884:     FastDisp ('        Es können so ev. Te
       ile',44,20,Chr(HilfText));
[5ZG]  885:     FastDisp ('        des Files unbrauchb
       ar',44,21,Chr(HilfText));
[L4H]  886:     FastDisp ('        überschrieben werde
       n!',44,22,Chr(HilfText));
[000]  887:
[61B]  888:     WindowOn (3,8,78,11,HilfRahmen,0,False)
       ;
[824]  889:     Window (1,1,80,25);
[8K9]  890:     FastDisp ('Ersetze:',5,10,Chr(Info));
[85F]  891:     WasErsetzen := SpezEin (14,10,30,Chr(In
       fo),WasErsetzen);
[000]  892:
[MKE]  893:     FastDisp ('Bis ans FileEnde   : _ (j/n)
       ',47,10,Chr(Info));
[GB5]  894:     If GanzesFile then
[8Z9]  895:      FastDisp ('J',68,10,Chr(EingabeFeld))
[A11]  896:     else
[UAA]  897:      FastDisp ('N',68,10,Chr(EingabeFeld));
[WK3]  898:     GotoXY (68,10);
[2V5]  899:     If Eingabe <> #27 then
[U05]  900:      Read (kbd,Eingabe);
[64B]  901:     If (Eingabe = #27) and not Keypressed t
       hen
[BG1]  902:     Begin
[WO4]  903:      WasSuchen := '';
[GK4]  904:      WasErsetzen := '';
```

```
[XZ2]  905:       WindowOff;
[721]  906:       EXIT;
[TUO]  907:      end;
[000]  908:
[1U4]  909:      If Eingabe = '' then
[8Y3]  910:       Eingabe := ' ';
[RLA]  911:      If (Eingabe = #27) and KeyPressed then
[BG1]  912:      Begin
[UO5]  913:        Read (kbd,Eingabe);
[8Y3]  914:        Eingabe := ' ';
[TUO]  915:      end;
[SZ7]  916:      Eingabe := UpCase (Eingabe);
[5G5]  917:      If Eingabe = 'J' then
[VV4]  918:       GanzesFile := True;
[1H5]  919:      If Eingabe = 'N' then
[P45]  920:       GanzesFile := False;
[XZ2]  921:      WindowOff;
[TUO]  922:    end;
[EL7]  923:    EinmalGefunden := False;
[MA9]  924:    Laenge := Length(WasErsetzen);
[SV2]  925:    Anzahl := 0;
[000]  926:
[X42]  927:    Repeat
[232]  928:      Finde (9);
[G85]  929:      If gefunden then
[BG1]  930:      Begin
[PR6]  931:        Anzahl := Succ(Anzahl);
[FC7]  932:        EinmalGefunden := True;
[BL2]  933:        WoBinIch;
[WV6]  934:        For Loop1 := 1 to Laenge do
[BG1]  935:        Begin
[SM9]  936:          WorkReal := (PosCursorFile-2)+Loop1
       ;
[CK8]  937:          If WorkReal < FileLaenge then
[BG1]  938:          Begin
[ZL8]  939:            Seek (FileEinEinzel,WorkReal);
[XGC]  940:            Write (FileEinEinzel,WasErsetzen
       [Loop1]);
[TUO]  941:          end;
[TUO]  942:        end;
[TUO]  943:      end;
[BEH]  944:    until (not gefunden) or (Keypressed) or (
       not GanzesFile);
[TVL]  945:    If (EinmalGefunden) and (GanzesFile) and
       (not keypressed) then
[BG1]  946:    Begin
[V1F]  947:      WindowOn (20,4,77,8,HilfRahmen,HilfHint
       erGrund,True);
[SZ5]  948:      Str (Anzahl:8,String8);
[MSA]  949:      FastDisp (String8,22,6,Chr(HilfText));
[3WS]  950:      FastDisp ('Zeichenfolgen wurden ersetzt
       ! (Taste drücken)',31,6,Chr(HilfText));
[UO5]  951:      Read (kbd,Eingabe);
[UW1]  952:      Warte;
[XZ2]  953:      WindowOff;
[TUO]  954:    end;
[T75]  955:    If KeyPressed then
[BG1]  956:    Begin
[COB]  957:      WindowOn (3,18,58,22,HilfRahmen,0,True)
       ;
[LCS]  958:      FastDisp ('Zeichenfolge ersetzen unterb
       rochen ! (Taste drücken)',5,20,Chr(Info));
[UW1]  959:      Warte;
[UO5]  960:      Read (kbd,Eingabe);
[XZ2]  961:      WindowOff;
[DLO]  962:    end
[A11]  963:    else
[DR7]  964:    If not EinmalGefunden then
[BG1]  965:    Begin
[5LG]  966:      WindowOn (3,18,39,22,HilfRahmen,FehlerH
       interGrund,True);
[Y2L]  967:      FastDisp ('Nichts gefunden ! (Taste drü
       cken)',5,20,Chr(FehlerText));
[UO5]  968:      Read (kbd,Eingabe);
[XZ2]  969:      WindowOff;
[TUO]  970:    end;
[W34]  971:    Window (2,2,79,24);
[NF2]  972:    ClrScr;
[544]  973:    BildSchirm (1);
[NT4]  974:    GanzesBildAus (0);
[824]  975:    Window (1,1,80,25);
[TUO]  976: end;
[000]  977:
[GEC]  978: { ----------------------------------------
       ---------------------------------- }
[000]  979:
[7V9]  980: Procedure EscNullEin (was : Char);
[000]  981:
[BG1]  982: Begin
[DMA]  983:    Seek (FileEinEinzel,PosCursorFile-1);
[LB7]  984:    If was = EscEingabe then
[BG1]  985:    Begin
[RO5]  986:      Eingabe := Chr(27);
```

```
[BE8]  987:      Write (FileEinEinzel,Eingabe);
[DLO]  988:    end
[A11]  989:    else
[BG1]  990:    Begin
[WV4]  991:      Eingabe := Chr(0);
[BE8]  992:      Write (FileEinEinzel,Eingabe);
[TUO]  993:    end;
[N4C]  994:    FastDisp (Eingabe,XPos,YPos,Chr(Daten));
[FK3]  995:    PfeilRechts;
[TUO]  996: end;
[000]  997:
[GEC]  998: { ----------------------------------------
       ---------------------------------- }
[000]  999:
[3B8] 1000: Procedure HilfeAusgeben;
[000] 1001:
[BG1] 1002: Begin
[SWF] 1003:    WindowOn (19,3,74,21,HilfRahmen,HilfHinte
       rGrund,True);
[DVP] 1004:    FastDisp ('File Utility V1.2          (c)
       1988 DOS International',21, 4,Chr(HilfText))
       ;
[9HC] 1005:    FastDisp ('----------------------------
       ----------------------',21, 5,Chr(HilfText))
       ;
[NSS] 1006:    FastDisp ('Pfeile = Bewegt den Cursors (a
       uf, ab, rechts, links)',21, 6,Chr(HilfText))
       ;
[6ZM] 1007:    FastDisp ('PgUp   = Blättert ein Bild zur
       ück.'                  ,21, 7,Chr(HilfText))
       ;
[63M] 1008:    FastDisp ('PgDn   = Blättert ein Bild vor
       .'                     ,21, 8,Chr(HilfText))
       ;
[YHM] 1009:    FastDisp ('Home   = Sprung an den FileAnf
       ang.'                  ,21, 9,Chr(HilfText))
       ;
[AHL] 1010:    FastDisp ('End    = Sprung an das FileEnd
       e.'                    ,21,10,Chr(HilfText))
       ;
[6EM] 1011:    FastDisp ('Esc    = Beendet das File Util
       ity.'                  ,21,11,Chr(HilfText))
       ;
[CLC] 1012:    FastDisp ('----------------------------
       ----------------------',21,12,Chr(HilfText))
       ;
[B7T] 1013:    FastDisp ('Cursor Sprünge innerhalb des a
       ktiven Bildes:'        ,21,13,Chr(HilfText))
       ;
[LRM] 1014:    FastDisp ('Crtl Home        = Cursor nach
       oben links.'           ,21,15,Chr(HilfText))
       ;
[83N] 1015:    FastDisp ('Ctrl PgUp        = Cursor nach
       oben rechts.'          ,21,16,Chr(HilfText))
       ;
[K2P] 1016:    FastDisp ('Ctrl PgDn        = Cursor nach
       unten rechts.'         ,21,17,Chr(HilfText))
       ;
[YEN] 1017:    FastDisp ('Ctrl End         = Cursor nach
       unten links.'          ,21,18,Chr(HilfText))
       ;
[PFR] 1018:    FastDisp ('Ctrl PfeilLinks  = Cursor nach
       Zeilen Anfang.'        ,21,19,Chr(HilfText))
       ;
[KWP] 1019:    FastDisp ('Ctrl PfeilRechts = Cursor nach
       Zeilen Ende.'          ,21,20,Chr(HilfText))
       ;
[000] 1020:
[UO5] 1021:    Read (kbd,Eingabe);
[UW1] 1022:    Warte;
[GFG] 1023:    WindowOn (22,7,77,24,HilfRahmen,HilfHinte
       rGrund,False);
[EWP] 1024:    FastDisp ('File Utility V1.2          (c)
       1988 DOS International',24, 8,Chr(HilfText))
       ;
[AKC] 1025:    FastDisp ('----------------------------
       ----------------------',24, 9,Chr(HilfText))
       ;
[RGF] 1026:    FastDisp ('F1       = Hilfe.'
                                 ,24,10,Chr(HilfText))
       ;
[T7L] 1027:    FastDisp ('F2       = Neues File anspreche
       n.'                    ,24,11,Chr(HilfText))
       ;
[4TL] 1028:    FastDisp ('F3       = Finde eine Zeichenfo
       lge.'                  ,24,12,Chr(HilfText))
       ;
[KAP] 1029:    FastDisp ('F4       = Auf File schreiben,
       Ein/Aus schalten.'     ,24,13,Chr(HilfText))
       ;
[9GP] 1030:    FastDisp ('F5       = Nochmals suchen nach
       Zeichenfolge von F3.' ,24,14,Chr(HilfText))
       ;
```

```
[TFM] 1031:    FastDisp ('F6       = Aktive Seite in AscI
          I anzeigen.'              ,24,15,Chr(HilfText))
           ;
[Y4R] 1032:    FastDisp ('F7       = Finden und ersetzen
          einer Zeichenfolge.'      ,24,16,Chr(HilfText))
           ;
[G8M] 1033:    FastDisp ('F8       = Aktive Seite in Hex
          anzeigen.'                ,24,17,Chr(HilfText))
           ;
[MON] 1034:    FastDisp ('F9       = Nochmals finden und
          ersetzen.'                ,24,18,Chr(HilfText))
           ;
[C5M] 1035:    FastDisp ('F10      = Code ESC in das File
           schreiben.'              ,24,20,Chr(HilfText))
           ;
[HUM] 1036:    FastDisp ('Alt-F10 = Code 0 in das File s
          chreiben.'                ,24,21,Chr(HilfText))
           ;
[6KN] 1037:    FastDisp ('Ctrl F1 = Dez, Hex, AscII Tabe
          lle anzeigen.'            ,24,23,Chr(HilfText))
           ;
[UO5] 1038:    Read (kbd,Eingabe);
[XZ2] 1039:    WindowOff;
[TUO] 1040: end;
[000] 1041:
[GEC] 1042: { ---------------------------------------
          -------------------------------- }
[000] 1043:
[BT6] 1044: Procedure TabelleAus;
[000] 1045:
[U66] 1046: Var    X,Y                 : Integer;
[000] 1047:
[BG1] 1048: Begin
[8Y3] 1049:    Eingabe := ' ';
[1WF] 1050:    WindowOn (40,3,70,23,HilfRahmen,HilfHinte
          rGrund,True);
[PUG] 1051:    FastDisp ('Dez Hex AscII Dez Hex AscII',4
          2,3,Chr(HilfRahmen));
[7R2] 1052:    X := 42; Y := 4;
[442] 1053:    Loop := 0;
[EU9] 1054:    While (Loop < 256) and (Eingabe <> #27) d
          o
[BG1] 1055:    Begin
[883] 1056:      Ordi := Loop;
[8U5] 1057:      Hexi := Trunc(Loop/16);
[CF5] 1058:      Str (Loop:3,String3);
[SMA] 1059:      FastDisp (String3,X,Y,Chr(HilfText));
[3KC] 1060:      FastDisp (HexBez [Hexi],X+5,Y,Chr(HexTe
          xt));
[C3F] 1061:      FastDisp (HexBez [ordi-(hexi*16)],X+6,Y
          ,Chr(HexText));
[LKB] 1062:      FastDisp (Chr(Loop),X+10,Y,Chr(HilfText
          ));
[YD3] 1063:      If X = 42 then
[KU9] 1064:       FastDisp ('|',X+13,Y,Chr(HilfText));
[9W2] 1065:      Y := Succ(Y);
[3M6] 1066:      If (Y = 23) and (X = 56) then
[BG1] 1067:      Begin
[UT4] 1068:        Read (Kbd,Eingabe);
[UW1] 1069:        Warte;
[NF2] 1070:        ClrScr;
[7R2] 1071:        X := 42; Y := 4;
[TUO] 1072:      end;
[3E3] 1073:      If Y = 23 then
[BG1] 1074:      Begin
[GD1] 1075:         Y := 4;
[KC1] 1076:         X := 56;
[TUO] 1077:      end;
[WP4] 1078:      Loop := Succ(Loop);
[TUO] 1079:    end;
[2V5] 1080:    If Eingabe <> #27 then
[UO5] 1081:     Read (kbd,Eingabe);
[XZ2] 1082:    WindowOff;
[TUO] 1083: end;
[000] 1084:
[GEC] 1085: { ---------------------------------------
          -------------------------------- }
[000] 1086:
[UPA] 1087: Procedure FileNeu (Woher:Integer);
[000] 1088:
[BG1] 1089: Begin
[XD4] 1090:    FileOn (Woher);
[UNB] 1091:    FileRecAnzahl := LongFileSize (FileEin);
[BAC] 1092:    FileLaenge := LongFileSize (FileEinEinzel
          );
[664] 1093:    PosRecord := 0;
[H84] 1094:    PosAnzeige := 1;
[032] 1095:    XPos := 2;
[132] 1096:    YPos := 2;
[DX4] 1097:    PosObenLinks := 1;
[NX7] 1098:    SegM := Seg (BildSpeicher[1]);
[EP8] 1099:    OfSegM := Ofs (BildSpeicher[1]);
[5S9] 1100:                              { String78 hat 78
          LeerZeichen !!! }
```

```
[R3C] 1101:    String78 := '
                           ';
[044] 1102:    BildSchirm (0);
[NT4] 1103:    GanzesBildAus (0);
[TUO] 1104: end;
[000] 1105:
[GEC] 1106: { ---------------------------------------
          -------------------------------- }
[000] 1107:
[BG1] 1108: Begin
[494] 1109:    SetCursor (0,12);
[W04] 1110:    WasSuchen := '';
[GK4] 1111:    WasErsetzen := '';
[Z25] 1112:    Schreiben := False;
[L58] 1113:    GrossBuchstaben := False;
[VV4] 1114:    GanzesFile := True;
[000] 1115:
[P73] 1116:    FileNeu (0);
[X52] 1117:    repeat
[MW4] 1118:      GotoXY (XPos,YPos);
[BL2] 1119:      WoBinIch;
[UW1] 1120:      Warte;
[NK4] 1121:      Read (Kbd,Befehl);
[1Z9] 1122:      If (Befehl = #27) and Keypressed then
[BG1] 1123:      Begin
[NK4] 1124:        Read (Kbd,Befehl);
[934] 1125:        Case Befehl of
[FR7] 1126:          Hilfe          : HilfeAusgeben;
[3R5] 1127:          Finden         : Finde (0);
[MN7] 1128:          NochmalsFinden : Finde (1);
[NS9] 1129:          FindenErsetzen : FindeErsetze (0);
[CR9] 1130:          NochmalsFindEr : FindeErsetze (1);
[TK7] 1131:          SchreibenEinAus: Begin
[VZ7] 1132:                             Schreiben := Not
          Schreiben;
[544] 1133:                             BildSchirm (1);
[TUO] 1134:                           end;
[D65] 1135:          Hoch           : Repeat
[D75] 1136:                             BufferLoeschen;
[KC3] 1137:                             PfeilAuf (1);
[BL2] 1138:                             WoBinIch;
[K86] 1139:                           until Not KeyPress
          ed;
[B96] 1140:          Links          : PfeilLinks;
[P87] 1141:          Rechts         : PfeilRechts;
[SV5] 1142:          Runter         : PfeilAb;
[KX6] 1143:          SeiteRunter    : Repeat
[D75] 1144:                             BufferLoeschen;
[961] 1145:                             PgDn;
[BL2] 1146:                             WoBinIch;
[K86] 1147:                           until Not KeyPress
          ed;
[OE6] 1148:          SeiteHoch      : Repeat
[D75] 1149:                             BufferLoeschen;
[481] 1150:                             PgUp;
[BL2] 1151:                             WoBinIch;
[K86] 1152:                           until Not KeyPress
          ed;
[3U5] 1153:          Anfang         : GoBegin;
[HD4] 1154:          Ende           : Begin
[2S1] 1155:                             GoEnd;
[OK2] 1156:                             RUnten;
[TUO] 1157:                           end;
[7W6] 1158:          NeuesFile      : FileNeu (1);
[P88] 1159:          AscIIAnzeigen  : AscIIAnzeige;
[YN7] 1160:          HexAnzeigen    : HexAnzeige;
[006] 1161:          LinksOben      : LOben;
[866] 1162:          RechtsOben     : ROben;
[7P6] 1163:          LinksUnten     : LUnten;
[NW6] 1164:          RechtsUnten    : RUnten;
[KM6] 1165:          ZeileAnfang    : ZAnfang;
[RX5] 1166:          ZeileEnde      : ZEnde;
[21F] 1167:          EscEingabe     : If Schreiben then
          EscNullEin (Befehl);
[D8F] 1168:          NullEingabe    : If Schreiben then
          EscNullEin (Befehl);
[F87] 1169:          Tabelle        : TabelleAus;
[TUO] 1170:        end;
[DLO] 1171:      end
[A11] 1172:      else
[VUA] 1173:      If (Befehl <> #27) and (Schreiben) then
[BG1] 1174:      Begin
[DMA] 1175:        Seek (FileEinEinzel,PosCursorFile-1);
[XMB] 1176:        FastDisp (Befehl,XPos,YPos,Chr(Daten)
          );
[FY7] 1177:        Write (FileEinEinzel,Befehl);
[FK3] 1178:        PfeilRechts;
[TUO] 1179:      end;
[KA4] 1180:    until Befehl = #27;
[824] 1181:    Window (1,1,80,25);
[NF2] 1182:    ClrScr;
[A74] 1183:    SetCursor (7,7);
[5TO] 1184: end.
```

Listing. Der Quellcode für »filed.pas«. Hercules-Benutzer bitte die Werte in Klammern eingeben.

Unsere Produkte finden Sie auch im Computer-, Buch- und Zeitschriften-Handel. Bitte achten Sie auf diesen Software-Ständer.

PC-SIG 499 PROCOMM **DM 16,--**
Das Kommunikationsprogramm schlechthin, vielleicht das Beste überhaupt. Benutzerfreundlich und leistungsstark. Vertrödeln Sie nicht Ihre Zeit mit schlechteren Programmen! Achten Sie aber unbedingt darauf, daß Sie nur die letzte Version von PROCOMM für Ihre Zwecke testen.

PC-SIG 5, 730 PC-File+ **DM 32,--**
Leistungsstarkes und sehr komfortables Datenverwaltungsprogramm. Menügeführte Anwahl aller Funktionen. Umfangreiche Hilfsbildschirme. Bildschirmorientierter Maskeneditor. Rechenfuntionen. Relationale Datenübergabe. Schnelle Such- und Sortierfunktionen. Paßwortschutz. Tastatur-Makros. Suchen Sie gleichzeitig in allen 70 Feldern. Bis zu 65533 Datensätze möglich. Im- und Export von Daten in 10 unterschiedlichen Formaten möglich. Ideal für die Verwaltung von Kundendaten, Artikeldateien, Vereinsdaten usw..

PC-SIG 728, 729 HOMEBASE **DM 32,--**
Speicherresidenter Desktop-Organizer der Spitzenklasse von BrownBagSoftware. Als echtes Universalgenie bietet HOMEBASE viele Funktionen, welche jederzeit aus anderen Programmen heraus aufgerufen werden können: Festplatten-Manager, Online Datenbank, Kommunikationsprogramm, mehrere Kalender, Taschenrechner, DOS Service, Adressbuch, Report Generator, Cut&Paste, Bildschirmuhr und noch mehr. Ein absolutes Spitzenprogramm ! ! !

PC-SIG 727 POWERMENÜ **DM 16,--**
PC-SIG 608 AUTOMENÜ **DM 16,--**
Diese beiden Menüsysteme sollten Sie unbedingt einmal ausprobieren, es sei denn, Sie verwenden bereits ein Menüsystem, mit dem Sie wirklich rundherum zufrieden sind. Sie werden es nicht bereuen! Diese Menüsysteme helfen Zeit und Nerven sparen. Ausführlicher Vergleichstest im PC-Magazin Nr. 47 vom November 1987 (Markt& Technik).

PC-SIG auch auf CD-ROM **DM 995,--**
Die CD-ROM enthält die ersten 817 PC-SIG Disketten. Treiber für Hitachi, Sony, Phillips und Sanyo sind im Preis enthalten. Ab Lager lieferbar!

PC-SIG 830, 831 WAMPUM **DM 32,--**
Umfassende und menügesteuerte Implementierung der dBaselll Programmiersprache. WAMPUM kann als Entwicklungssystem und als Anwenderdatenbank genutzt werden. Trotz sehr einfacher Handhabung extrem leistungsstark, schnell und vorallem kompatibel zum dBaselll-Standard. 80 Seite Dokumentation auf Diskette vorhanden. 320K freier Arbeitsspeicher erforderlich.

PC-SIG 696 QEBECALC **DM 16,--**
Das leistungsstärkste und anwenderfreundlichste Tabellenkalkulationsprogramm in der PC-SIG Library. Geeignet für zwei- und dreidimensionale Rechenblätter mit bis zu 260000 Zellen. Auf Knopfdruck werden die Werte eines Arbeitsblattes als Balkengrafik dargestellt. Der Datenwürfel kann in verschiedene Lagen gedreht werden, dadurch ist eine ausgezeichnete Darstellung aller vorhandenen Daten gewährleistet. Alle wünschenswerten Funktionen aus Mathematik, Trigonometrie, Statistik usw. sind vorhanden, daher geeignet für alle Anwendungen.

PC-SIG 836 DISK COMMANDO **DM 16,--**
Das ist die lange erwartete Shareware-Antwort auf die NORTON UTILITIES. Enthält 17 unentbehrliche Utilities für die Arbeit mit Dateien auf Diskette und Festplatte. Z.B. Optimieren des Diskspeicherplatzes und dadurch mehr Platz und kürzere Zugriffszeiten, Wiederherstellung von versehentlich gelöschten Dateien, Zurückholen von Daten nach ungewollter Formatierung der Disk, Disktester usw.. In der Shareware-Version sind die Auswahlmenüs mit blinkenden Dollarzeichen eingerahmt.

PC-SIG 873 WINDOW BOSS **DM 16,--**
POP-UP WINDOWS und PULLDOWN MENÜS für den ernsthaften C-Programmierer (alle wichtigen C-Compiler werden berücksichtigt).

PC-SIG 885 MISCELLANEOUS UTILITIES **DM 16,--**
Unentbehrliches Hilfsmittel für alle Programmierer
- C Window Toolbox
- 52 Utility-Routinen für TURBO PASCAL
- Konvertierung von TURBO PASCAL Source in C Source
- 200 Funktionen in C und Assembler.

PC-SIG 893 WEAK LINK **DM 16,--**
.Nach Verbindung von zwei IBM-kompatiblen Rechnern über die serielle Schnittstelle lassen sich die Diskettenlaufwerke (auch Festplatte) des anderen Rechners genauso benutzen, wie die am Master installierten Laufwerke. D.h. einfache Übertragung von Dateien und Dateigruppen von PC zu PC mittels COPY-Befehl, besonders nützlich bei Rechnern mit unterschiedlichem Diskettenformat (5 1/4" nach 3 1/2"). Ebenfalls ideal für Systemhäuser zum Installieren von vielen und großen Dateien direkt von Festplatte zu Festplatte. Natürlich auch für Backup-Zwecke bestens geeignet. Sehr hohe Übertragungsgeschwindigkeit (bis zu 115200 baud). Einfache Handhabung, da gute und ausführliche Dokumentation auf Diskette mitgeliefert wird. Sourcecode in der registrierten Version verfügbar.

PC-SIG INFORMATIONSPAKET **DM 10,--**
enthält folgende Bestandteile:
1. Wissenswertes zu Shareware und PUBLIC DOMAIN
2. TOP 100+ (engl. Kurzbeschreibung der Bestseller, 30 Seiten)
3. Ein Ansichtsexemplar PC-SIG MAGAZINE
4. Unkosten für Verpackung und Versand

Da die PC-SIG Library inzwischen auf mehr als 1000 Disketten mit Software aus allen Bereichen (Textverarbeitung, Datenbanken, Kommunikationsprogramme, Programmiersprachen, Grafikprogramme, Utilities, Benutzeroberflächen, Spiele, Lernsoftware usw.) angewachsen ist, können wir leider eine umfassende Information nicht mehr kostenlos abgeben. Eine Kurzinformation auf einigen Blättern Papier nützt bei dieser Fülle an Software nur sehr wenig.
Wirklich umfassende und fortlaufende Information ist ohnehin nur in Form einer PC-SIG Mitgliedschaft möglich. Wenn Sie bisher von PC-SIG oder Shareware noch nicht viel gehört haben, so gibt Ihnen dieses INFORMATIONSPAKET Auskunft über alle wichtigen Fragen in Verbindung mit Freier Software, Shareware und PUBLIC DOMAIN Software. Sie erhalten außerdem eine Übersicht über die besten Programme in der PC-SIG Library.

PC-SIG Mitgliedschaft (kein Kaufzwang) **DM 68,--**
- aktuelle Beschreibung der gesamtem PC-SIG Library 1-705 gedruckt auf 430 Seiten, Rest auf Diskette
- zweimonatliche Zusendung des PC-SIG MAGAZINES
- zweimonatliche Zusendung des PC-SIG HOT SHEETS
- ab 5 Disketten Mitgliedsrabatt von DM 2,-- pro Disk
- Sonderverkaufsaktionen nur für PC-SIG Mitglieder

Schicken Sie noch heute den Scheck über DM 68,-- für Ihre einjährige PC-SIG Mitgliedschaft oder wählen Sie:

PC-SIG Mitgliedschaft + 4 PC-SIG Disketten Ihrer Wahl **DM 98,--**
Mitgliedschaft wie oben beschrieben und vier beliebige Programmdisketten aus der PC-SIG Library.

5 Stck. **Katalog**-Disketten mit kompl. Beschreibung **je DM 10,--**
1-200 / 201-400 / 401-600 / 601-800 / 801-1000

Komplette Beschreibung der PC-SIG Disketten 1-705 gedruckt als Buch mit ca. 430 Seiten, gegliedert nach Sachgebieten, **sehr empfehlenswert** **DM 28,--**

multigraph II V3.0 **DM 112,--**
Das speicherresidente Programm ermöglicht CGA-Grafik auf einer Hercules-Monochrom-Grafikkarte. Endlich Grafik auch mit GW-Basic und TURBO PASCAL auf PCs mit HERCULES-Karte. CGA-Programme laufen ohne Zeitverzögerung. Einfache Handhabung. Umfangreiche deutschsprachige Beschreibung (36 Seiten) ist im Preis enthalten. Multigraph bewältigt auch selbststartende Spiele wie z.B. Flight Simulator von MS, Nightmission Flipper, Summer Games, Winter Games, Decathlon, Tapper, usw..
Besitzer von älteren Versionen von Multigraph II erhalten nach Einsendung der Originaldiskette für nur DM 10,-- Unkostenbeitrag die letzte Version. Update zu Multigraph I DM 25,--.

multigraph 1640 **DM 98,--**
Spezialversion für den SCHNEIDER PC1640 MD.

Kaufen Sie auch zu Ihrer Sicherheit nur die grauen Original-PC-SIG Disketten vom autorisierten PC-SIG Händler. Sie kaufen Qualität zum günstigen Preis und unterstützen gleichzeitig die Arbeit von PC-SIG. Nur so gibt es auch weiterhin jeden Monat ca. 20 neue Disketten und zusätzlich noch 20 Updates zu früheren Nummern.

Original PC-SIG Disketten erhalten Sie z.B. bei:

Günter Abbé
Meginhardstr. 30, 8031 Gilching
Tel. 08105/9284

Pan Tau Data GmbH
Dachauer Str. 300, 8000 München 50
Tel. 089/1495119

Computerhaus Gerhard Klaus
Lange Str. 131, 2850 Bremerhaven
Tel. 0471/58986

PC-Service Gunter Eibl
Feldstr. 18, 8012 Ottobrunn
Tel. 089/6091920

Eckhard H. Bragard
Automaten-Electronic-EDV
Bierstr. 141, 5120 Herzogenrath 1
Tel. 02406/2316

Computer Markt München GmbH
Perlacher Str.19-21, 8000 München 90
Tel. 089/695969

Klaus Weiß GmbH
Rathausplatz, 8094 Edling
Tel. 08071/2472

Elektonik-Vertrieb Karl-Heinz Harter
Salmstr. 13, 7550 Rastatt 15
Tel. 07229/2133

Lips Computer- und Softwaremarkt
Ingolstädter Str. 58L, 8000 München 45
Tel. 089/3189090

Ingenieurbüro Klaus Bühler
Schafstr. 16, 7440 Nürtingen
Tel. 07022/33137

Bücher-Balogh
Sandstr. 1, 5900 Siegen
Tel. 0271/55298

PDC
Luisenstr. 115, 6380 Bad Homburg
Tel. 06172/24748

ITC. Import Export GmbH
Stresemannstr. 50, 2800 Bremen
Tel. 0421/498111

Peksoft
Müllerstr. 44, 8000 München 5
Tel. 089/2604674

Computer- und Medienberatung Reinhard Schrutzki
Lorichstr. 6, 2000 Hamburg 60
Tel. 040/6306262

Kaufhaus COM
Computer & Technologie Vertrieb
Westenrieder Str. 32, 8000 München2
Tel. 089/292066

ZEITLUPE, deutschsprachiges Utility **DM 16,--**
Dieses speicherresidente Programm bremst die Arbeitsgeschwindigkeit Ihres PCs. Unbedingt notwendig, wenn man für XT geschriebene Programme (z.B. Spiele) auf AT oder 8 MHz XTs laufen lassen will. Ideal für Spiele wie Defender of the Crown. Auch eine lange Winternacht geht einmal zu Ende. Überlisten Sie den Computer durch dieses nicht ganz faire Hilfsmittel. Bremsen Sie mit ZEITLUPE die Arbeitsgeschwindigkeit Ihres Rechners nur dann, wenn es langsamer gehen soll. D.h.Laden von der Diskette und andere zeitintensive Operationen werden nicht gebremst.

PC-WRITE V2.71, engl. **DM 48,--**
(Achtung: PC-Write ist nur bis zur Version V2.4 Shareware. Alle höheren Versionen und die Übersetzungen dürfen nicht kopiert und weitergegeben werden!)

Dieses Angebot enthält 2 lizensierte Disketten, ein Quick Guide (40 Seiten) und das sehr nützliche PC-Write Tutorial (16 Seiten). Alles original vom Programmautor !! Das Original-Hardcover-Manual mit ca. 350 Seiten einschl. dem Laser Printer Guide können wir zum Preis von nur DM 78,-- liefern.

PC-WRITE gehört zu den weltweit am meisten eingesetzten Textverarbeitungsprogrammen. Kein anderes Programm bietet ein besseres Preis-Leistungs-Verhältnis. PC-Write ist für den Anfänger genauso geeignet wie für den Profi. Ideal einsetzbar zum Schreiben von einfachen Briefen und Handzetteln, besonders empfehlenswert aber zum Erstellen von umfangreichen Dokumentationen, wissenschaftlichen Arbeiten und Büchern. Automatisches Erstellen von Inhalts- und Stichwortverzeichnis. Automatische Indexverwaltung. Serienbriefe erstellen, ändern und drucken. Gleichzeitige Bearbeitung von mehreren Texten. Kinderleichte Bedienung. Umfangr. Hilfsmenüs durch Tastendruck zu aktivieren. Blitzschnelles Scrollen und Blättern am Bildschirm, daher ausgezeichnet geeignet für Mausbedienung. Komfortable Makrofunktion. Blitzschnelles suchen und ersetzen von Wörtern und Zeichen. Stromausfallsicherung durch automatische Speicherung zwischendurch. Freie Belegung fast aller Tasten mit beliebig langen Strings. PC-Write erstellt ASCII-Dateien, daher keine Probleme mit fremden Dateien. PC-Write unterstützt alle Drucker !!!

Lassen auch Sie sich von PC-Write begeistern. Wegen der wesentlich schnelleren Arbeit mit Mausunterstützung ist das Programm besonders geeignet für den Einsatz auf SCHNEIDER PCs.

PC-Write ist auch deutschsprachig verfügbar. Einzelheiten und Preise bitte anfragen.

Bitte beachten Sie, daß auf allen Disketten aus der PC-SIG Library sowohl Programme, wie auch Textdateien nur englischsprachig sind. Deutschsprachige Anleitungen sind für einige Programme verfügbar. Auslieferung in der Regel im 5 1/4" MS(PC)-DOS Diskettenformat. Aufpreis DM 4,-- je Diskette für 3 1/2"-Format, wobei dann Zweiersets auf einer Diskette ausgeliefert werden. Günstige Staffelpreise für Mitglieder und bei Abnahme von größeren Stückzahlen. Bitte nur Kleinbeträge bis maximal DM 20,-- als Schein oder in Briefmarken einsenden. Lieferung nur gegen Nachnahme oder Vorauskasse, außer bei Behörden, Großfirmen und nach vorheriger Absprache. Bestellungen mit Vorausscheck werden vorrangig behandelt. Bitte Versandkostenanteil von DM 4,-- bei Bestellungen bis DM 150,-- berücksichtigen.

PC-SIG AUTHORIZED DEALER

Kirschbaum Software GmbH

Kronau 15, D-8091 Emmering
Tel. 08067/1220 oder 1051, Fax. 08067/1053

Distributor in der Schweiz:
DSE Datasystems-Engineering AG, Badener Str. 262, CH 8004 Zürich
Tel. 01/2418882, Telex 813950

Distributor in Österreich:
Überreuter Media Handels- u. Verlagsges. mbH, Laudongasse 29, A-1082 Wien
Tel. 0222/481543, Fax. 433313

So sieht Ihre Datei wirklich aus

»Vis« macht die »unsichtbaren« Zeichen einer Datei sichtbar. Dazu werden die entsprechenden Werte in darstellbare Codes umgewandelt. Eine nützliche C-Utility für jedermann.

Textverarbeitungsprogramme speichern die Texte in aller Regel mit besonderen Codes für Unterstreichen, Fettschrift und ähnlichem. Die Codierungen bestehen zum Teil aus Escape-Sequenzen, die sich weder am Bildschirm noch auf einem Drucker ausgeben lassen. Der Ausdruck solcher Dateien mit »type« führt fast immer zu Ausgaben, die nicht lesbar sind. Die Escape-Zeichen werden nämlich auch zur Steuerung des Bildschirms und des Druckers benutzt.

Mit dem MS-DOS-Programm »debug« können Sie sich natürlich die Daten ansehen, das Ergebnis ist jedoch wenig befriedigend, da bei der Ausgabe mit »debug« alle Zeichen (auch die druckbaren) hexadezimal wiedergegeben werden. Auch die Zeilenstruktur der Texte bleibt nicht erhalten. »debug« und andere Dump-Programme sind bei der Anwendung auf Textdateien sehr unübersichtlich – und damit für diese Aufgabe nicht geeignet.

»Vis« gibt demgegenüber nur die nicht druckbaren Zeichen einer Textdatei in einer wählbaren Codierung wieder, es behält auch die Zeilenstruktur bei. »Vis« kann damit als ein »Semi-Filedump« bezeichnet werden.

In der Funktion »main()« werden nacheinander alle beim Programmaufruf angegebenen Argumente getestet. Die Funktion läßt außer den Optionen keine weiteren Argumente zu. Die Angaben dürfen in Klein- oder in Großbuchstaben geschrieben werden. Sie werden mit den Zeichen » – ae- oder »/« eingeleitet. Das Programm hält sich damit sowohl an den Unix- als auch den MS-DOS-Standard. Folgende Optionen stehen Ihnen zur Auswahl:

? oder H – Anzeige einer Kurzanleitung. Wird diese Option gewählt, bricht das Programm nach Ausgabe des Hilfstextes ab.

S – Alle nicht druckbaren Zeichen werden getilgt. Die Steuerzeichen »Tab« und Zeilenwechsel bleiben uncodiert erhalten.

V – Alle nicht druckbaren Zeichen werden codiert, also auch »Tab« und Zeilenwechsel (Zeilenwechsel = »\n«)

O – Codierung in Oktalschreibweise

D – Codierung in Dezimalschreibweise

C – Codierung durch Kontrollzeichen

Bei Angabe eines unzulässigen Arguments oder einer fehlerhaften Option bricht das Programm mit einer Fehlermeldung ab. Der Exit-Code des Programms ist in diesem Fall größer Null.

Nachdem alle Programmparameter gesetzt sind, wird die Funktion »work()« aufgerufen, die den eigentlichen Kern dieser Utility bildet. Die Funktion »work()« liest Zeile für Zeile vom Standard-Eingabegerät und codiert jedes eingelesene Zeichen gemäß den gewählten Optionen.

Der Befehl »Vis« hat folgende Syntax:

```
vis [-H ?] [-S] [-V] [-O] [-D] [-C] [<Datei1] [>Datei2 ]
```

»Vis« liest die Eingaben des Standard-Eingabegeräts, entschlüsselt die nicht druckbaren Zeichen und setzt Sie in einen druckbaren Code um. Zum Schluß wird die Eingabe auf dem Standard-Ausgabegerät ausgegeben. Mit Hilfe der Ein-/Ausgabe-Umleitung liest »Vis« die Datei »Datei1« und schreibt die Daten in die Datei »Datei2«. Als druckbare Zeichen akzeptiert »Vis« alle Zeichen mit den ASCII-Werten zwischen 32 und 127. Den erweiterten IBM-Zeichensatz kann das Programm nicht sichtbar machen.

Je nach gewählter Funktion werden die nicht druckbaren Zeichen in hexadezimaler, oktaler oder dezimaler Schreibweise wiedergegeben. Ferner können die Zeichen auch als C-Escape-Sequenz oder Kontrollzeichen angezeigt werden.

Der Seitenvorschub (ASCII 12) wird in den verschiedenen Varianten wie folgt dargestellt:

hexadezimal \0C
oktal \014
dezimal \d12
C-Escape \f
Ctrl-Zeichen \↑L

Die hexadezimale und oktale Schreibweise sowie die C-Escape-Sequenzen entsprechen der »C-Notation«. Die dezimale Darstellung und die Kontrollzeichen sind dem angepaßt. Geben Sie beim Programmaufruf keine Option an, so nimmt »Vis« folgende Codierung vor:

- »Tab« und Zeilenwechsel werden nicht codiert
- alle anderen nicht druckbaren Zeichen werden in hexadezimaler Schreibweise oder als C-Escape-Sequenz wiedergegeben
- das Zeichen »\« selbst wird in »\\« umgewandelt

Folgende Beispiele zeigen Ihnen die Arbeitsweise des Vis-Befehls:

```
vis > text1
```

»Vis« leitet alle Eingaben über die Tastatur in die Datei »text1« um. Die Eingabe muß mit [Ctrl-Z] beendet werden. Nicht druckbare Zeichen wandelt »Vis« in C-Escape-Sequenzen oder in hexadezimale Schreibweise um.

```
vis -o <text1 >text2
```

Die Datei »text1« wird zeilenweise gelesen. Nicht druckbare Zeichen werden in C-Escape-Sequenzen oder oktale Schreibweise umgewandelt und in »text2« ausgegeben.

```
vis -s <text1
```

Die Datei »text1« wird gelesen, wobei alle nicht druckbaren Zeichen außer »Tab« und Zeilenwechsel unterdrückt werden.

Im Programmteil »main()« (Listing 1) wird in einer While()-Schleife – beginnend bei Zeile 86 – die beim Programmaufruf angegebene Kommandozeile abgearbeitet. Innerhalb dieser Schleife verzweigt die Switch()-Anweisung entsprechend dem gefundenen Optionsbuchstaben:

```
switch (c=*++cmdlarg, c=toupper(c))
{
...
...
}
```

Die Anweisungen innerhalb der Klammer haben folgende Bedeutung:

- Char-Pointer »cmdlarg« inkrementieren
- Options-Buchstaben in Großbuchstaben umwandeln
- Options-Buchstaben der Variablen »c« zuweisen

Setzt man diese Anweisungen in die C-Befehlsfolge um, so hat man sich einen typischen C-Fallstrick in das Programm eingebaut:

```
switch (c=toupper(*++cmdlarg))
{
...
...
}
```

Der Befehl »toupper()« ist nicht als Funktion, sondern als Makro in der Header-Datei »ctype.h« implementiert:

```
#define toupper(a) ( (islower(a)) ? toupper(a) : (a) )
```

Nach Auflösung der Makros durch den Preprozessor erhält man für die Anweisung »c = toupper(* + + cmdlarg)« die aufgelöste Befehlsfolge

```
c=( (islower(*++cmdlarg)) ? _toupper(*++cmdlarg) :
(*++cmdlarg) )
```

bei welcher der Zeiger »cmdlarg« zweimal inkrementiert wird. In diesem Fall wird der Variablen »c« nicht mehr der Optionsbuchstabe zugewiesen, sondern lediglich nutzlose Informationen aus dem dahinter folgenden Speicher. Durch die Anweisungsfolge »c = * + + cmdlarg, c = toupper(c))« kann dieser Effekt umgangen werden.

Innerhalb eines C-Programms dürfen zusammenhängende Texte mit aufeinanderfolgenden Printf()-Anweisungen ausgegeben werden. Hier stellt »Vis« für den Programmierer eine wesentliche Hilfe dar. Die Funktion »Hilfe()« im Quellcode »vis.c« (Zeile 219) wurde bei der Entwicklung des Programms zuerst als Leerfunktion geschrieben: »Hilfe(){}«.

Der Text, der beim Aufruf von »vis -?« angezeigt wird, wurde dann mit einem Editor erfaßt und als ASCII-Datei in der Datei »help.txt« abgelegt. Ansi-Escape-Sequenzen zur Steuerung der Bildschirmausgabe wurden unmittelbar in diesen Text mit eingegeben.

Mit dem Hilfsprogramm »ptic.c« (Print Text In C)

```
# include <stdio.h>
# define MAXCHAR 255
main()
{
char Zeile[MAXCHAR];
while (gets(Zeile) != NULL)
printf("printf(\"%s\");\n", Zeile);
}
```

können Textzeilen einer ASCII-Textdatei als C-Source-Code

```
printf("Textzeile 1");
printf("Textzeile 2");
...
...
```

in eine neue Textdatei ausgegeben werden, die wiederum als Include-Datei benutzt werden kann. Zur endgültigen Implementierung der Funktion »Hilfe()« in unserem Programm »Vis« kann nun die Datei »help.txt« mit der Befehlsfolge

```
vis -q <help.txt | ptic >vishelp.i
```

in die Include-Datei »vishelp.i« (Listing 2) umgewandelt werden.

Der Befehl »vis -q« bewirkt dabei, daß alle nicht druckbaren Zeichen wie Tabulatoren, Escape-Sequenzen und Zeilenwechsel in C-Notation ausgegeben werden. Die Funktion »Hilfe()« in »vis.c« ist anschließend durch die Anweisung

```
# include "vishelp.i"
```

zu ergänzen. Nach erneuter Compilierung von »vis.c« steht die Online-Programmhilfe zur Verfügung. Die mühsame Eingabe der Printf()-Anweisungen und der C-Escape-Sequenzen entfällt damit. Auf diese Weise können Texte, die beliebige Steueranweisungen (zum Beispiel für den Drucker) enthalten, in C-Quellcode umgewandelt werden. Der Arbeitsgang ist immer der gleiche:

- Text mit Editor erfassen
- Text formatiert in Zwischendatei ausgeben
- Zwischendatei mit »vis -q« filtern
- und mit »ptic« in C-Source-Code umwandeln

»Vis« läßt sich für vielfältige Zwecke einsetzen und ist somit eine wirklich nützliche Ergänzung für Ihre Systemdiskette.

(Manfred Beer/sp)

```
[ESD]    0: /* *** Programm vis.c ********************
           ********************************** */
[9V8]    1: /*
                                                 */
[R6P]    2: /*  VIS : Nicht druckbare Zeichen in Textda
           teien sichtbar machen                 */
[9V8]    3: /*
                                                 */
[Y7E]    4: /*  Betriebssystem :  MS-DOS ab 2.00
                                                 */
[9V8]    5: /*
                                                 */
[N4D]    6: /*  Compiler       :  MICROSOFT C ab 3.00
                                                 */
[9V8]    7: /*
                                                 */
[X7E]    8: /*       Compilieren..............  : >msc vi
           s;                                    */
[6WD]    9: /*       Linken...................  : >link v
           is;                                   */
[1W9]   10: /*       oder
                                                 */
[16G]   11: /*       Compilieren und Linken..  : >cl vis
           ;                                     */
[9V8]   12: /*
                                                 */
[9V8]   13: /*
                                                 */
[E9C]   14: /*  Compiler       :  TURBO C ab 1.00
                                                 */
[9V8]   15: /*
                                                 */
[GHK]   16: /*       Compilieren und Linken..  : >tcc -m
           t -w- -f vis                          */
[LXK]   17: /*       ggf. umwandeln .com-File  : >exe2bi
           n vis.exe vis.com                     */
[9V8]   18: /*
                                                 */
[ALH]   19: /*       Compiler-Optionen.......  : -mt
           Model...Tiny                          */
[93E]   20: /*                                     -w-
           Display warnigs...Off                 */
[N1E]   21: /*                                     -f-
           Floating point...None                 */
[9V8]   22: /*
                                                 */
[LEC]   23: /*  Programmaufruf :
```

```
                                                 */
[YSF]   24: /*       >vis [Optinen] [<Datei1] [>Datei2]
                                                 */
[W9C]   25: /*  Programmhilfe  :                  */
[VXA]   26: /*       >vis [-H!-?]
                                                 */
[9V8]   27: /*
                                                 */
[1GC]   28: /*                    (s.u.: Funktion Help()
           )                                     */
[9V8]   29: /*
                                                 */
[MFH]   30: /*  Copyright (C)  : ing.-buero beer, ettli
           ngen                                  */
[9V8]   31: /*
                                                 */
[VCB]   32: /* ************************************
           ********************************** */
[000]   33:
[SU5]   34: # include <stdio.h>
[7Y5]   35: # include <ctype.h>
[000]   36:
[0T7]   37: # define OFF             0       /* 00000000
           */
[FX8]   38: # define VERBOSE_ON      1       /* 00000001
           */
[UU8]   39: # define STRIP_ON        2       /* 00000010
           */
[BR8]   40: # define OCTAL_ON        8       /* 00001000
           */
[SM8]   41: # define DECI_ON        16       /* 00010000
           */
[9S8]   42: # define CTRL_ON        32       /* 00100000
           */
[000]   43:
[BC5]   44: # define SAPCE           ' '
[3E5]   45: # define TAB             '\t'
[L66]   46: # define FORMFEED        '\f'
[000]   47:
[H4D]   48: # define OPT_ERR         1       /* fehlerha
           ft Option */
[ADG]   49: # define CMDL_ERR        2       /* Fehler i
           n der Kommandozeile */
[000]   50:
[000]   51:
[1R6]   52: /* Pseudofunktionen */
```

```
[000]   53:
[R5E]   54: # define SETWORKFLAG(a)   workflag = workfl
          ag | a
[000]   55:
[000]   56:
[L37]   57: /* *** globale Variblen *** */
[000]   58:
[DN8]   59:          unsigned char workflag = OFF;
[000]   60:
[Z9C]   61:          int  ctrl_char[]={'\n'    ,  '\t',
          '\f', '\0',  -1};
[DFD]   62:          char *ctrl_str[]={"\\n\n", "\\t", "
          \\f", "\\0"};
[000]   63:
[XG9]   64:          char  opt_char[]={'S'      , 'V'
          , 'O'      ,
[6P3]   65:                                'D'      , 'C'
          , '\0'};
[Y3D]   66:          char  flag_val[]={STRIP_ON, VERBOSE
          _ON, OCTAL_ON,
[ZR4]   67:                                DECI_ON,  CTRL_ON
          };
[000]   68:
[NZ9]   69: /* *** Vorwaertsdeklarierungen *** */
[000]   70:
[RAK]   71:          int work_text();
          /* Arbeitsroutine fuer Textdateien */
[VOE]   72:          int Fehler();
                       /* Fehlerverwaltung */
[WZC]   73:          int Hilfe();
                         /* Programmhilfe */
[000]   74:
[000]   75:
[000]   76:
[000]   77:
[ZFG]   78: /*=======================================
          ==================================*/
[NG4]   79:    main (argc, argv)
[ZFG]   80: /*=======================================
          ==================================*/
[YN2]   81:    int argc;
[EL2]   82:    char **argv;
[V30]   83: {
[VZ3]   84:          char *cmdlarg;
[095]   85:          register char c;
[ENH]   86:          while   (cmdlarg = *++argv)
                   /* Kommandozeile abarbeiten */
[V30]   87:                  {
[AM9]   88:                  if      (*cmdlarg == '-' ||
          *cmdlarg == '/')
[BAA]   89:                          switch  (c=*++cmdla
          rg, c = toupper(c))
[VYO]   90:                                  {
[AA5]   91:                                  register ch
          ar i;
[9Z1]   92:                                  case '?' :
[6U4]   93:                                  case 'H' :
            Hilfe();
[7V1]   94:
            exit(0);
[000]   95:
[NG6]   96:                                  default  :

[Z38]   97:                                          for
          (i=0; opt_char[i] != '\0';++i)
[246]   98:
          if (opt_char[i] == c)
[V30]   99:
            {
[AA7]  100:
            SETWORKFLAG(flag_val[i]);
[KV1]  101:
            break;
[X30]  102:
            }
[2P6]  103:                                          if
          (opt_char[i] == '\0')
[V30]  104:
            {
[EA7]  105:
            Fehler(OPT_ERR, cmdlarg);
[FV3]  106:
            exit (OPT_ERR);
[X30]  107:
            }
[RM7]  108:                                  }
                       /* Ende switch () */
[A11]  109:                  else
[V30]  110:                          {
[9K7]  111:                                  Fehler(CMDL
          _ERR, cmdlarg);
[314]  112:                                  exit(CMDL_E
          RR);
[X30]  113:                          }
[789]  114:                  }
                         /* Ende while() */
[SE3]  115:          work_text();
[27C]  116:          exit(0);
                     /* Exit-Code setzen */
[X30]  117: }
```

```
[000]  118:
[000]  119:
[Z8C]  120: /*--------------------------------------
          ----------------------------------*/
[ETH]  121:    prctrl(c)                              /
          * nicht druckbare Zeichen ausgeben */
[Z8C]  122: /*--------------------------------------
          ----------------------------------*/
[KL1]  123:    int c;
[V30]  124: {
[168]  125:          if (workflag & CTRL_ON && c < SAPCE
          )
[V30]  126:          {
[WH5]  127:                  printf("\\^%c", c+'@');
[RP2]  128:                  return;
[X30]  129:          }
[PA6]  130:          if (workflag & OCTAL_ON)
[YS4]  131:                  printf("\\%03o", c);
[VD7]  132:          else  if (workflag & DECI_ON)
[T85]  133:                          printf("\\d%03d", c
          );
[A11]  134:                  else
[V75]  135:                          printf("\\x%02X", c
          );
[X30]  136: }
[000]  137:
[000]  138:
[Z8C]  139: /*--------------------------------------
          ----------------------------------*/
[XSE]  140:    work_text()
                     /* VIS Arbeitsrouitne */
[Z8C]  141: /*--------------------------------------
          ----------------------------------*/
[V30]  142: {
[KL1]  143:    int c;
[BS6]  144:    while ((c=getchar()) != EOF )
[V30]  145:          {
[ZX7]  146:          if (isascii(c) && isprint(c) )
[V30]  147:            {
[A89]  148:            if ( (c=='\\') && !(workflag & STRI
          P_ON))
[C34]  149:                    putchar('\\');
[M83]  150:          putchar(c);
[479]  151:          }
                                 /* Ende if */
[5Z9]  152:        else
                                  /* else 1 */
[V30]  153:          {
[BC6]  154:          if (workflag & STRIP_ON)
[OL2]  155:                  switch (c)
[V30]  156:                  {
[N92]  157:                  case '\n' :
[N92]  158:                  case '\t' :
[M83]  159:                          putchar(c);
[KV1]  160:                          break;
[ZC9]  161:                  }
                        /* Ende switch */
[6N9]  162:          else
                                   /* else 2*/
[V30]  163:            {
[2G6]  164:            if (workflag & VERBOSE_ON)
[V30]  165:                  {
[E45]  166:                  register int i;
[V58]  167:                  for(i=0;ctrl_char[i] != -1;
          ++i)
[CE6]  168:                          if (ctrl_char[i] ==
          c)
[KV1]  169:                                  break;
[NE6]  170:                  if (ctrl_char[i] != -1)
[Y56]  171:                          printf(ctrl_str[i])
          ;
[A11]  172:                  else
[6P2]  173:                          prctrl(c);
[4B8]  174:                  }
                            /* Ende if */
[VM9]  175:            else
                                  /* else 3 */
[V30]  176:                  {
[E45]  177:                  register int i;
[X08]  178:                  for(i=0;ctrl_char[i] != '\0
          '; ++i)
[CE6]  179:                          if (ctrl_char[i] ==
          c)
[KV1]  180:                                  break;
[LS6]  181:                  if (ctrl_char[i] != '\0')
[M83]  182:                          putchar(c);
[A11]  183:                  else
[6P2]  184:                          prctrl(c);
[6V8]  185:                  }
                        /* Ende else 3 */
[RR9]  186:            }
                        /* Ende else 2 */
[6R9]  187:          }
                        /* Ende else 3 */
[7SA]  188:    }
                       /* Ende while() */
[X30]  189: }
[000]  190:
[000]  191:
[Z8C]  192: /*--------------------------------------
          ----------------------------------*/
```

```
[TSK]  193:    int Fehler(FehlerNr, FehlerText)
                          /* Fehlermeldungen */
[Z8C]  194: /*--------------------------------------
            ----------------------------------*/
[RP3]  195:      int FehlerNr;
[045]  196:      char *FehlerText;
[V30]  197: {
[L45]  198:    switch (FehlerNr)
[V30]  199:    {
[ZE3]  200:      case OPT_ERR :
[NSL]  201:           fprintf (stderr, "vis : fehlerhaft
           e Option : %s\n", FehlerText);
[K22]  202:           break;
[000]  203:
[GV3]  204:      case CMDL_ERR :
[KPN]  205:           fprintf (stderr, "vis : Fehler in
           der Kommandozeile : %s\n", FehlerText);
[K22]  206:           break;
[000]  207:
[XZ3]  208:      default      :
[4LN]  209:           fprintf (stderr, "vis : unbekannte
            Fehlermeldung : %d\n", FehlerNr);
[GDD]  210:           fprintf (stderr, "      Programmab
           bruch!\n");
[8T2]  211:           exit(-1);
[000]  212:
[71D]  213:    }
                     /* Ende switch(FehlerNr) */
[W7C]  214: }
                     /* *** Ende Fehler() *** */
[000]  215:
[000]  216:
[000]  217:
[Z8C]  218: /*--------------------------------------
            ----------------------------------*/
[DUF]  219:    Hilfe()
               /* Anzeigen einer Programmhilfe */
[Z8C]  220: /*--------------------------------------
            ----------------------------------*/
[V30]  221: {
[YC6]  222: # include "vishelp.i"
[X30]  223: }
```

Listing 1. Der C-Quellcode für das Programm »Vis«

```
[5OP]    0: printf("VIS : Nicht druckbare Zeichen einer
            Textdatei sichtbar machen\n");
[TTE]    1: printf("      (C) ingenieurbuero beer, ettl
           ingen\n");
[GB3]    2: printf("\n");
[F1G]    3: printf("Aufruf\t     : >vis [Option] [I/O U
           mleitung]\n");
[GB3]    4: printf("\n");
[F7W]    5: printf("Beschreibung : vis liest zeichenwei
           se Text vom standart Eingebageraet, ersetzt\
            n");
[P2S]    6: printf("\t       alle nicht druckbaren Zeic
           hen durch  einen  druckbaren Code und\n");
[WBS]    7: printf("\t       und gibt die Zeichen auf d
           em standart Ausgabegeraet wieder aus.\n");
[GB3]    8: printf("\n");
[EAS]    9: printf("\t       Mit Hilfe der I/O Umleitun
           g kann aus einer Plattendatei gelesen\n");
[Z3T]   10: printf("\t       und in eine solche geschri
           eben werden. vis kann auch als Filter\n");
[2KD]   11: printf("\t       benutzt werden (piping).\n
           ");
[GB3]   12: printf("\n");
[LSK]   13: printf("Optionen     :\t-?!-H\tAusgabe dies
           er Programmhilfe\n");
[MZL]   14: printf("\t\t-S\talle nicht druckbaren Zeich
           en unterdruecken\n");
[6ZN]   15: printf("\t\t-V\tvollstaendige Umformung nic
           ht druckbarer Zeichen\n");
[P2B]   16: printf("\t\t-O\toktale Codierung\n");
[S8C]   17: printf("\t\t-D\tdezimale Codierung\n");
[18E]   18: printf("\t\t-C\tCodierung mit Ctrl-Zeichen\
           n");
[GB3]   19: printf("\n");
```

Listing 2. Der Hilfstext befindet sich in der Include-Datei »vishelp.i«

»Head« schafft Übersicht

»Head« dient zum schnellen Durchblättern aller Textdateien auf einer Diskette. Diese in C programmierte Funktion ist dem gleichnamigen Unix-Befehl nachempfunden.

Waren es bis vor einiger Zeit noch CP/M-Funktionen, die bei Programmierern hoch im Kurs standen, so sind es nun Unix-Funktionen wie »Make« und »Touch«, die unter MS-DOS nachgebildet werden. Auch »Head« zählt zu diesen von Unix her bekannten Kommandos.

Wohl jeder, der mit MS-DOS arbeitet, hat irgendwann einmal versucht mit »type *.bat«, »type *.txt« oder ähnlichen Befehlen eine Datei, deren Name spurlos aus dem Gedächtnis entschwunden war, wiederzufinden. Solche Versuche quittiert MS-DOS allerdings mit einer Fehlermeldung, da Wildcards und Joker im Type-Befehl nicht erlaubt sind. Aber selbst wenn dies funktionieren würde, ist es meist unnötig sich die gesamte Datei anzusehen, da schon wenige Zeilen genügen, die Datei zu identifizieren.

Abhilfe schafft die Unix-Funktion »Head«. »Head« zeigt die ersten Zeilen einer Datei an. Komfortabel wird die Funktion jedoch erst, wenn Wildcards und Joker in Argumenten der Befehlszeile akzeptiert werden. Bei dem Datalight Compiler, muß hierzu beispielsweise beim Linken die Funktion »__main.obj« mit eingebunden werden, andere C-Compiler besitzen ähnliche Routinen (bei Unix ist dies Standard).

Aufgerufen wird die Funktion mit

```
head <-zeilen> datei(1)...datei(n)
```

wobei die optionale Angabe »-zeilen« für die Zahl der auszugebenden Zeilen je Datei steht. Ohne diese Angabe ist hier der Wert 10 voreingestellt.

Der Programmablauf der Head-Funktion erklärt sich fast von selbst (Listing). Zuerst ermittelt das Progamm die gewünschte Zahl der auszugebenden Zeilen. Der Einfachheit halber wird das Switchzeichen » – ae- als Minuszeichen aufgefaßt, so daß mit dem Absolutwert des ersten Arguments die korrekte Zeilenanzahl bereitsteht. Die Variable »a«, die als Zeiger auf das nächste Argument fungiert, wird in diesem Fall erhöht und zeigt nun auf den nächsten Dateinamen. Ist kein weiterer Dateiname vorhanden, sollte der Benutzer über den korrekten Aufruf der Funktion aufgeklärt werden. Das Programm bricht in dem Fall nach der Ausgabe der Kurzanleitung mit »Errorlevel 1« ab.

Bei Compilern, die als erstes Argument nicht den eigenen Programmnamen liefern, muß die Variable »a« mit dem Wert Null initialisiert werden. Damit das Programm bei mehreren Dateien auch weiß, welche Datei gerade in Bearbeitung ist, wird bei mehr als einem verbleibenden Argu-

ment ein Schalter gesetzt, der später die Ausgabe des Dateinamens veranlaßt.

Die folgende Schleife leistet die Hauptarbeit: Zuerst öffnet »Head« die betreffende Datei. Sollte das nicht gelingen, so wird eine Fehlermeldung ausgegeben. Ist der oben erwähnte Schalter gesetzt, zeigt »Head« den Dateinamen an und gibt die gewünschte Anzahl von Zeilen der Datei aus. Zum Schluß wird die Datei geschlossen und die Routine beginnt von vorne – solange, bis alle Dateinamen abgearbeitet sind.

Der Befehl »printf« leitet die Ausgabe auf andere Geräte, wie beispielsweise den Drucker oder in ein anderes Programm, um. Benutzen Sie hier das MS-DOS-Programm »more«, so können Sie die Ausgaben in Ruhe lesen. Ebenso läßt sich durch die Ausgabe-Umleitung in eine Datei ein Versionsverzeichnis Ihrer Programme anlegen. Dazu schreiben Sie sinnvollerweise in der ersten Zeile jedes Programms als Kommentar Version und Datum.

Nach Eingabe der Befehlszeile

```
head -1 *.c > c_vers.txt
```

steht das gewünschte Verzeichnis in der Datei »c_vers.txt«.

Noch einen Vorteil gibt es gegenüber der Unix-Version: Während das Unix-Head nach dem ersten falschen Dateinamen abbricht, setzt das hier vorgestellte Programm die Ausgabe nach einer Fehlermeldung mit dem nächsten Dateinamen fort.

(Peter Koch/sp)

```
[F2B]    0: /** HEAD   Version 1.0   15.04.87   Peter K
           och **/
[LDE]    1: /* HEAD gibt die Anfangszeilen der angegebe
           nen  */
[G97]    2: /*    Dateien aus.
              */
[AVD]    3: /* Aufruf: head [-zeilen] datei(1)...datei(
           n)   */
[45E]    4: /*  wobei: zeilen   = Anzahl der auszugeben
           den  */
[9H6]    5: /*                    Zeilen
              */
[2H9]    6: /*          datei(x) = Dateinamen
              */
[XO7]    7: /*****************************************
           ******/
[000]    8:
[TV5]    9: #include <stdio.h>
[000]   10:
[974]   11: #define USAGE  \
[DXE]   12:  "\nAufruf: head [-zeilen] datei(1)...datei
           (n)\n"
[7Z3]   13: #define TRUE 1
[RB4]   14: #define FALSE 0
[000]   15:
[HV1]   16: FILE *fn;
[BL2]   17: char *file;
[000]   18:
[B65]   19: main(argc, argv)
[YN2]   20: int argc;
[OB3]   21: char *argv[];
[V30]   22: {
[NB7]   23: int a, b, zeilen, anzeige;
[YG4]   24: char zeile[132];
[F11]   25: a = 1;
[000]   26:
[KSE]   27: /* Wenn im Aufruf eine Zeilenanzahl angegeb
           en   */
[UCF]   28: /*  wurde, diese übernehmen, sonst als Stan
           dard */
[VXA]   29: /*  10 Zeilen pro Datei ausgeben.
              */
[000]   30:
[M45]   31: if (argv[a][0] == '-')
[XR8]   32:    zeilen = abs(atoi(argv[a++]));
[A11]   33: else
[UW2]   34:    zeilen = 10;
[000]   35:
[POF]   36: /* Wurde kein Dateiname angegeben, Ausgabe
           der  */
[3EA]   37: /*  'Bedienungsanleitung'.
              */
[000]   38:
[G14]   39: if (argc - a == 0)
[V30]   40:    {
[377]   41:    fprintf(stderr, USAGE);
[DV1]   42:    exit(1);
[X30]   43:    }
[000]   44:
[D6D]   45: /* Bei mehr als einer Datei, Schalter setze
           n,   */
[SEC]   46: /*  zur Ausgabe des Dateinamens.
              */
[000]   47:
[2CA]   48: anzeige = (argc - 1 > 1) ? TRUE : FALSE;
```

```
[000]   49:
[AKD]   50: /* Ab hier erfolgt die Abarbeitung der Date
           i-   */
[M5F]   51: /*  namen bis zum bitteren Ende und Ausgabe
           der */
[DE8]   52: /*  x-ersten Zeilen.
              */
[000]   53:
[DX3]   54: for (; a<argc; ++a)
[V30]   55:    {
[W54]   56:    file = argv[a];
[000]   57:
[ZKD]   58:    /* Öffnen der Datei, bzw. Fehlermeldung,
              */
[XDC]   59:    /*  wenn das nicht funktioniert.
              */
[000]   60:
[VA8]   61:    if ((fn = fopen(file, "r")) == NULL)
[XG5]   62:       fprintf(stderr,
[XHA]   63:          "\nDatei %s nicht gefunden\n", fil
           e);
[A11]   64:    else
[V30]   65:       {
[000]   66:
[YCD]   67:       /* Ausgabe des Dateinamens falls Scha
           lter */
[ZN6]   68:       /*  gesetzt.
              */
[000]   69:
[E83]   70:       if (anzeige)
[RY7]   71:          printf("\n==> %s <==\n", file);
[000]   72:
[E9C]   73:       /* Datei lesen und Ausgabe der Zeile
           bis  */
[YWB]   74:       /*  EOF oder Anzahl der auszugebenden
              */
[1B8]   75:       /*  Zeilen erreicht.
              */
[000]   76:
[5P8]   77:       for (b = 1; fgets(zeile, 132, fn) &&
[DR3]   78:                                 b <= zeilen;
           ++b)
[485]   79:          printf("%s", zeile);
[000]   80:
[LDC]   81:       /* Schließen der Datei. (Die Fehlerme
           l-   */
[2PB]   82:       /*  dung nur der Ordnung halber)
              */
[000]   83:
[PL4]   84:       if (fclose(fn) == 1)
[XG5]   85:          fprintf(stderr,
[TV9]   86:             "\nFehler bei CLOSE von %s\n",
           file);
[X30]   87:       }
[X30]   88:    }
[X30]   89: }
```

Listing. Der Quellcode des C-Programmes »Head«

»Move«

– Verschiebung ohne Bewegung

Das in C geschriebene Programm »Move« verschiebt Dateien auf einem Datenträger ohne den Nachteil des MS-DOS-Befehls »Copy«. Es bleibt nämlich keine Datei an der alten Stelle zurück.

Wer kennt nicht die folgende Situation: Eine Datei steht in einen Unterverzeichnis, wäre aber in einem ganz anderen bedeutend besser aufgehoben. MS-DOS bietet zwei Befehle zur Lösung dieses Problems. Sie kopieren die Datei mit dem Befehl

```
..\copy \subdir1\datei.ext \subdir2
```

von Verzeichnis 1 in Verzeichnis 2 und löschen mit

```
del \subdir1\datei.ext
```

die ursprüngliche Datei im ersten Verzeichnis. Alle Bytes von »datei.ext« werden dabei in den Systemspeicher und von dort wieder zurück auf den Massenspeicher transportiert. Der Vorgang nimmt bei größeren Dateien natürlich einige Zeit in Anspruch. Außerdem entsteht an der ursprünglichen Position der Datei auf dem Datenträger ein sogenanntes »Loch«, das durch das Löschen einer Datei für die Wiederverwendung freigegeben ist. Werden solche Vorgänge oft wiederholt, so kommt das Betriebssystem »ganz schön ins Schwitzen«, denn falls für eine Datei, die größer als das größte »Loch« ist, Speicherplatz gesucht wird, muß MS-DOS die Datei auf mehrere dieser Freistellen aufteilen.

Ein C-Programm macht »Move« zu einem leistungstarken DOS-Befehl

Um den durch die »Dateizerstückelung« immer langsamer werdenden Datenzugriff wieder auf Trab zu bringen, bieten sich spezielle Programme wie »Soft-Logic Solutions Disk Optimizer« oder »Nortons Speed Disk« an. Diese Programme reorganisieren den Datenträger wieder. Im Klartext heißt das, daß die verstreuten Teile einer einzelnen Datei wieder zusammengesucht und in einem Stück abgelegt werden. Nach getaner Arbeit bleibt ein einziger zusammenhängender freier Datenträgerbereich über.
Nach diesen Ausführungen verstehen Sie sicher, daß hier ein Befehl, der die Dateien an ihrem ursprünglichen Platz auf dem Datenträger läßt und nur die Directory-Einträge verändert, weitaus bessere Dienste leistet. Das abgedruckte C-Programm »Move« (Listing) erledigt diese Arbeit schnell und zuverlässig.
»Move« basiert auf »Function Call 56H« (Change Directory Entry) des DOS-Interrupt 21H. Im Handbuch »MS-DOS 3.1 Programmers's Reference Manual« ist nachzulesen, daß diese Funktion in »DS:DX« einen Zeiger auf die Zeichenkette der Quelldatei und in »ES:DI« einen Zeiger auf die Zeichenkette der Zieldatei erwartet. Leider ist es nicht möglich, bei dieser Funktion mit sogenannten »Wildcards« zu arbeiten. »copy« und andere MS-DOS-Befehle gestatten hingegen die Arbeit mit den Symbolen »?« und »*«, um zum Beispiel alle Dateien mit der Extension ».pas« (*.pas) in eine andere Directory zu kopieren.
Mit einem kleinen Assemblerprogramm für »Function Call 56H« wäre das Verschieben einer einzigen Datei durchaus zu realisieren. Für mehrere Dateien müßte das Programm entsprechend oft aufgerufen – oder aber eine entsprechende Batch-Datei angelegt werden.
Mehr Komfort bekommen Sie allerdings, wenn die Behandlung von Dateigruppensymbolen (Wildcards) in das Move-Programm integriert wird. Der weitaus größte Teil des Quellcodes von »Move« ist für die Erkennung und Behandlung dieser Symbole zuständig. Die Syntax von »Move« entspricht dabei dem Copy-Befehl. Mit

```
move \dir1 \dir2
```

werden zum Beispiel alle Dateien aus »Dir1« mit gleichem Namen in »Dir2« eingetragen. Directory 1 ist anschließend leer. Der Befehl

```
move \dir1\*.txt \dir2\*.bak
```

trägt alle Dateien aus Directory 1 mit der Extension ».txt« in Directory 2 mit der Erweiterung ».bak« ein.
Abgesehen davon, daß die Dateien nicht wirklich kopiert, sondern lediglich die Directory-Einträge verändert werden, gibt es noch einen weiteren Unterschied zwischen »Move« und »Copy«. Für den Fall, daß im Ziel-Directory bereits Datei gleichen Namens existiert, wird die Routine mit der Fehlermeldung »Access denied« abgebrochen. »Copy« überschreibt dagegen jede Datei ohne Rückmeldung an den Benutzer.
Prinzipiell ließe sich der Move-Befehl auch an Stelle von »Rename« einsetzen. Allerdings gibt das nur wenig Sinn, denn als nichtresidentes Kommando muß das Programm »move.com« oder »move.exe« mit einer Länge von zirka 8 KByte erst einmal geladen werden, bevor die Befehlsausführung erfolgt.
Bei Durchsicht des Listings erkennen Sie, daß auf besondere Funktionen bestimmter C-Compiler verzichtet wurde. Das Programm kann daher mit allen gängigen C-Compilern (Turbo C, Microsoft C, Quick C) mit dem Speichermodell »Small« übersetzt werden. (Frank Mersmann/sp)

```
[ZHA]    0:  /*****************************************
             ********************************\
[YZ7]    1:  *                                        *
[A4L]    2:  *  Programm.......:  Move - Dateien ohne B
             ewegung verschieben                      *
[PKB]    3:  *  Hauptfile......:  Move.C
                                                      *
[F8B]    4:  *  Link OBJ-Files.:  ---
                                                      *
[3MH]    5:  *  Compiler-Model.:  Small, z.B. Turbo C 1
             .0, Microsoft C 4.0, ...                 *
[FLD]    6:  *  Copyright......:  F. Mersmann, 19.07.87
                                                      *
[4YB]    7:  *  Version, Stand.:  1.24, 06.08.87
                                                      *
[YZ7]    8:  *                                        *
[KGA]    9:  \*****************************************
             ********************************/
```

```
[000]   10:
[340]   11: /*
[WZM]   12:  *  Move "Quelle" "Ziel" wirkt wie: COPY "Q
             uelle" nach "Ziel", DEL "Quelle".
[LGR]   13:  *  Bei Move wird aber das Quellfile nicht
             wirklich transportiert, sondern
[W6N]   14:  *  nur der DIR-Eintrag geändert. Dies spar
             t Zeit und 'zerstückelt' die
[6BA]   15:  *  Festplatte oder Diskette nicht.
[840]   16:  */
[000]   17:
[AT5]   18: #include "ctype.h"
[DS5]   19: #include "stdio.h"
[406]   20: #include "string.h"
[S65]   21: #include "dos.h"
[000]   22:
[2C5]   23: #define  FALSE          0
[K65]   24: #define  TRUE           1
[UN9]   25: #define  Version        "move [1.24]"
[T26]   26: #define  Beep           0x07
```

```
[T1H]   27: #define  Setdta         0x1A            /*
           Function Calls zu DOS INT 21 */
[CR6]   28: #define  First          0x4E
[T86]   29: #define  Next           0x4F
[F27]   30: #define  cdEntry        0x56
[TSK]   31: #define  PathLen        40              /*
           kann an eigene Bedürfnisse ange-
[2D4]   32:
           passt werden */
[000]   33:
[P38]   34: /* --- Typendeklarationen --- */
[OCG]   35: struct  DTA {                          /* Buffer D
           isk Transfer Address */
[DAF]   36:           char  res[21];               /* siehe MS
           -DOS Programmer's Ref. Man. */
[F82]   37:           char  att;
[9C4]   38:           int   time, date;
[KY2]   39:           long  fsize;
[WTB]   40:           char  fname[14];             /* aktuelle
           r Filename */
[LO1]   41:         };
[000]   42:
[000]   43:
[737]   44: /* --- Globale Variable --- */
[CVE]   45: int             fcount;                /*
           Anzahl gemoveter Files */
[BVG]   46: union   REGS    inregs, outregs;       /*
           CPU-Regs fuer DOS-Calls */
[RZ6]   47: struct  SREGS   segmregs;
[2PF]   48: struct  DTA     dtabuffer;             /*
           Buffer für DOS Calls */
[000]   49:
[000]   50:
[1AA]   51:                 /* ---- Unterprogramme fuer
           Move ---- */
[E06]   52:                 /* ------------------------
           ---------- */
[000]   53:
[WR9]   54: /*  --- Unterprogramm: disperror ---
[BHF]   55:  *      Fehlermeldungen entsprechend errorn
           r ausgeben
[840]   56:  */
[72A]   57: int     disperror (errdcl, errornr)
[AD4]   58: char    *errdcl;
[8D4]   59: int     errornr;
[000]   60:
[W85]   61: { char  *errstr;
[000]   62:
[3Y5]   63:   switch (errornr)  {
[VUE]   64:     case  2: errstr = "File(s) not found\n"
           ;              break;
[EHE]   65:     case  3: errstr = "Path not found\n";
                         break;
[34E]   66:     case  5: errstr = "Access denied\n";
                         break;
[H6B]   67:     default: errstr = "Not same device";
[333]   68:   }/*switch*/
[1BP]   69:   fprintf (stderr, "%s error: '%s' %s%c", V
           ersion, errdcl, errstr, Beep);
[F84]   70: }/*disperror*/
[000]   71:
[000]   72:
[6Z9]   73: /*  --- Unterprogramm: printusage ---
[B1A]   74:  *      Benutzungshinweise ausgeben
[840]   75:  */
[BE5]   76: void    printusage ()
[V30]   77: {
[5X5]   78:   fprintf (stderr,
[X1N]   79:     "%s usage: [d:][path] sourcefile [d:][
           path][destfile]\n", Version);
[5X5]   80:   fprintf (stderr,
[PSE]   81:     "                    DOS wildcards '?'
           and / or '*' are allowed\n");
[3T4]   82: }/*printusage*/
[000]   83:
[000]   84:
[CL9]   85: /*  --- Unterprogramm: showexit ---
[RYH]   86:  *      Anzahl 'gemoveter' Files anzeigen u
           nd Programm beenden.
[840]   87:  */
[YU4]   88: void    showexit ()
[V30]   89: {
[U7C]   90:   printf ("\n%11d File(s) moved\n", fcount)
           ;
[VV1]   91:   exit (0);
[1U3]   92: }/*showexit*/
[000]   93:
[000]   94:
[D38]   95: /*  --- Unterprogramm: parse ----
[17H]   96:  *      "instr" in Path, Filenamen und File
           Extension zerlegen.
[RCK]   97:  *      Wenn keine Extension vorhanden ist,
           letzten Stringteil
[UCB]   98:  *      erstmal als File interpretieren.
[840]   99:  */
```

```
[GPA]  100: void    parse (instr, pname, fname, fext)
[ZG6]  101: char    *instr, *pname,
[294]  102:         *fname, *fext;
[000]  103:
[3M2]  104: { char  *p,
[DT1]  105:         *temp,
[TGG]  106:         tempstr[PathLen];               /*
           Platz für Kopie von instr */
[000]  107:
[3V8]  108:   temp = strcpy (tempstr, instr);
[000]  109:
[4LC]  110:   /* Extension abtrennen falls vorhanden. *
           /
[GP9]  111:   if (NULL != (p = strrchr (temp, '.'))) {
[KV4]  112:     strcpy (fext, p);
[Z49]  113:     *p = '\0';                            /*
           sonst 0 setzen */
[XRO]  114:   }
[FZ3]  115:   else *fext = '\0';
[000]  116:
[1DG]  117:   /* suche '\' oder ':' um DIR und/oder Dri
           ve abzutrennen */
[YC3]  118:   *pname = '\0';
[LRH]  119:   if (NULL != (p = strrchr (temp, '\\')) ||
           NULL != (p = strchr (temp, ':'))) {
[FN9]  120:     strncat (pname, temp, (++p - temp));
[EV1]  121:     temp = p;
[X30]  122:   }
[000]  123:
[FNA]  124:   /* Der Rest ist erstmal Filename */
[RL5]  125:   strcpy (fname, temp);
[RU2]  126: }/*parse*/
[000]  127:
[000]  128:
[4U9]  129: /*  --- Unterprogramm: patchdest ----
[CSL]  130:  *      "dststr" entsprechend Wildcards und
           "srcstr" veraendern. Wenn
[AAL]  131:  *      keine Wildcards vorhanden sind, "ds
           tstr" unverändert lassen.
[840]  132:  */
[ZY9]  133: void    patchdest (srcstr, dststr)
[3V7]  134: char    srcstr[], dststr[];
[000]  135:
[D95]  136: { register int  i;
[000]  137:
[L57]  138:   /* finde 1ten Wildcard */
[KH4]  139:   i = 0;
[CHF]  140:   while (dststr[i] && dststr[i] != '*' && d
           ststr[i] != '?') i++;
[000]  141:
[TLM]  142:   /*  entsprechend "srcstr" Veränderungen a
           n "dststr" vornehmen.
[SOF]  143:    *  ? durch Char und * durch Zeichenkette
            ersetzen.
[840]  144:    */
[3D8]  145:   while (i < strlen (dststr)) {
[546]  146:     switch (dststr[i]) {
[X7H]  147:       case '?' : dststr[i] = i <= strlen (s
           rcstr) ? srcstr[i] : '\0';
[KV1]  148:                  break;
[CV8]  149:       case '*' : if (i <= strlen (srcstr))
[2A0]  150:                    do
[L47]  151:                      dststr[i] = srcstr[i];
[BF5]  152:                    while (srcstr[i++]);
[KV1]  153:                  break;
[333]  154:     }/*switch*/
[CHO]  155:     i++;
[FT2]  156:   }/*while*/
[E54]  157: }/*patchdest*/
[000]  158:
[000]  159:
[K7A]  160: /*  --- Unterprogramm: moveonefile ---
[YWK]  161:  *      Source aus Path und von DOS geliefe
           rtem DIR-Eintrag zusammen-
[VML]  162:  *      bauen. Wenn Dest-Extension vorhande
           n, Destination-String zu-
[36K]  163:  *      sammenfügen. Sonst mit Source-DIR-E
           intrag versuchen. 1 File
[ETH]  164:  *      'moven'. Evtl. Fehler anzeigen und
           abbrechen, sonst Erfolg
[2T3]  165:  *      anzeigen.
[840]  166:  */
[GPE]  167: void    moveonefile (spath, destp, destf, d
           este)
[967]  168: char    spath[], destp[],
[ZH5]  169:         destf[], deste[];
[000]  170:
[D95]  171: { register int  i;
[REF]  172:   int           dirflag,                 /*
           FALSE = destf kann nicht Teil  */
[FAE]  173:                 errnr;                   /*
           von DIR sein, sondern ist File */
[YC7]  174:   char          srcstr[PathLen],
[8N5]  175:                 dststr[PathLen];
[000]  176:
```

```
[SDM]  177:    /* Source aus Path und von DOS gelieferte
           m Filenamen zusammenbauen */
[M08]  178:    strcpy (srcstr, spath);
[1PA]  179:    strcat (srcstr, dtabuffer.fname);
[000]  180:
[3L9]  181:    /* Destination zusammenbauen */
[L38]  182:    strcpy (dststr, destp);
[S97]  183:    strcat (dststr, destf);
[000]  184:
[A9L]  185:    if (strlen (deste)) {                    /*
           normaler Filename mit Extension */
[VR4]  186:      dirflag = FALSE;
[L97]  187:      strcat (dststr, deste);
[B1K]  188:    }                /* FileString kann DIR od
           er File (es gab keine EXT) sein */
[3GK]  189:    else {                                   /* FileStri
           ng erstmal als DIR ausprobieren */
[WR4]  190:      dirflag = TRUE;
[42F]  191:      if (strlen (dststr) && dststr[strlen (d
           ststr)] != '\\')
[N76]  192:        strcat (dststr, "\\");
[7RA]  193:      strcat (dststr, dtabuffer.fname);
[X30]  194:    }
[000]  195:
[FUP]  196:    /* Fileeintrag 'umhängen'. Wenn Fehler, n
           och einmal versuchen. Jetzt aber */
[5FG]  197:    if (0 != (errnr = chdirentry (srcstr, dst
           str)) && dirflag) {
[LE7]  198:      strcpy (dststr, destp);
[M9L]  199:      strcat (dststr, destf);
       /* destf nicht als DIR einsetzen */
[AWA]  200:      errnr = chdirentry (srcstr, dststr);
[X30]  201:    }
[000]  202:
[883]  203:    if (errnr) {
[DH8]  204:      disperror (dststr, errnr);
[3PF]  205:      showexit ();                           /
           * Bei Fehler Programm beenden */
[X30]  206:    }
[K8G]  207:    else {                                  /
           * Erfolgreichen 'move' anzeigen */
[55G]  208:      i = 0;                 while (0 != (srcstr
           [i++] = toupper (srcstr[i])));
[EWF]  209:      i = 0;                 while (dststr[i++]
           = toupper (dststr[i]));
[17B]  210:      printf ("%-36s  -->  %s\n" ,srcstr, dst
           str);
[X30]  211:    }
[D35]  212: }/*moveonefile*/
[000]  213:
[000]  214:
[7M9]  215: /*  --- Unterprogramm: setdtabuf ---
[DX8]  216:  *      Disk Transfer Address setzen
[840]  217:  */
[RG6]  218: void    setdtabuf (dta)
[MV4]  219: struct  DTA     *dta;
[000]  220:
[V30]  221: {
[717]  222:   bdos (Setdta, FP_OFF (dta), 0);
[U44]  223: }/*setdtabuf*/
[000]  224:
[000]  225:
[1P9]  226: /*  --- Unterprogramm: findfirst ---
[LRA]  227:  *      1ten Dir-Eintag mit "name" finden
[840]  228:  */
[UZ7]  229: int     findfirst (name, att)
[PC3]  230: char    *name;
[K72]  231: int     att;
[000]  232:
[V30]  233: {
[925]  234:   inregs.h.ah = First;
[PR6]  235:   inregs.x.dx = FP_OFF (name);
[4M4]  236:   inregs.x.cx = att;
[PH8]  237:   intdos (&inregs, &outregs);
[LBC]  238:   return outregs.x.cflag ? outregs.x.ax : 0
       ;
[G54]  239: }/*findfirst*/
[000]  240:
[000]  241:
[HK9]  242: /*  --- Unterprogramm: findnext ---
[D89]  243:  *      Nächsten Dir-Eintrag finden
[840]  244:  */
[AH4]  245: int     findnext ()
[V30]  246: {
[KR4]  247:   inregs.h.ah = Next;
[PH8]  248:   intdos (&inregs, &outregs);
[LBC]  249:   return outregs.x.cflag ? outregs.x.ax : 0
       ;
[KM3]  250: }/*findnext*/
[000]  251:
[000]  252:
[6Z9]  253: /*  --- Unterprogramm: chdirentry ---
[5H3]  254:  *      do the 'move'
[840]  255:  */
```

```
[T89]  256: int     chdirentry (sname, dname)
[BE6]  257: char    *sname, *dname;
[000]  258:
[V30]  259: {
[SB6]  260:   inregs.h.ah = cdEntry;
[T76]  261:   segread (&segmregs);
[G57]  262:   inregs.x.dx = FP_OFF (sname);
[1M7]  263:   segmregs.es = segmregs.ds;
[U07]  264:   inregs.x.di = FP_OFF (dname);
[HUC]  265:   intdosx (&inregs, &outregs, &segmregs);
[LBC]  266:   return outregs.x.cflag ? outregs.x.ax : 0
       ;
[BT4]  267: }/*chdirentry*/
[000]  268:
[000]  269:
[000]  270:
[PU6]  271: /*  --- Hauptprogramm ---  */
[NG4]  272: main (argc, argv)
[EG3]  273: int             argc;
[R55]  274: char            *argv[];
[000]  275:
[549]  276: { /* Lokale Variable von main () */
[MK3]  277:   int           errnr;
[NSE]  278:   char          dpath[PathLen],
                /* Destination Path */
[S9D]  279:                 dfname[PathLen],
                /* Destination File */
[3AD]  280:                 dfext[5],
                /* Destination Extension */
[PFB]  281:                 spath[PathLen],
                /* Source Path */
[CXB]  282:                 sfname[PathLen],
                /* Source File */
[YPB]  283:                 sfext[5],
               /* Source Extension */
[3YC]  284:                 argv1[PathLen],
                /* Kopie von argv[1] */
[L9E]  285:                 argv2[PathLen];
                /* wenn argv[2] nicht da */
[000]  286:
[AN3]  287:   printf ("\n");
[7HF]  288:   if (argc < 2 )  {
                /* Command Line in Ordnung? */
[TW7]  289:     printusage ();  showexit ();
[X30]  290:   }
[NZF]  291:   else argc == 2 ? argv2[0] = '\0' : strcpy
          (argv2, argv[2]);
[000]  292:
[NXK]  293:   /* zerlege Source und Dest-Name in Path,
          File und Extension */
[HU9]  294:   parse (argv[1], spath, sfname, sfext);
[369]  295:   parse (argv2,   dpath, dfname, dfext);
[000]  296:
[57F]  297:   setdtabuf (&dtabuffer);
                /* DTA einrichten */
[WT2]  298:   fcount = 0;
[3DH]  299:   strcpy (argv1, argv[1]);
                /* Kopie für Veränderungen */
[000]  300:
[WAF]  301:   if (errnr = findfirst (argv1, 0))
                /* gibt es 1 File? */
[4Z5]  302:     if (! strlen (sfext)) {
[LCF]  303:       strcat (argv1, "\\*.*");
                /* ganze Directory moven? */
[NF8]  304:       errnr = findfirst (argv1, 0);
[X30]  305:     }
[000]  306:
[883]  307:   if (errnr) {
[NWG]  308:     errnr == 18 ? disperror (argv1, 2) : di
          sperror (argv1, errnr);
[BF3]  309:     showexit ();
[X30]  310:   }
[S2D]  311:   else parse (argv1, spath, sfname, sfext);
[000]  312:
[FL8]  313:   /* es gibt mindestens 1 File */
[EWO]  314:   do {
[YZF]  315:     strcpy (argv1, spath);
                /* Source erneut zerlegen */
[4D9]  316:     strcat (argv1, dtabuffer.fname);
[PC9]  317:     parse (argv1, spath, sfname, sfext);
[5PH]  318:     patchdest (sfname, dfname);
                /* evtl. in Dest. Wilcards */
[8VF]  319:     patchdest (sfext, dfext);
                /* ersetzen                */
[TWC]  320:     moveonefile (spath, dpath, dfname, dfex
          t);
[LM2]  321:     fcount++;
[HXK]  322:     parse (argv2, dpath, dfname, dfext); }
                /* erforderl. falls Wildc. */
[BV8]  323:   while (! (errnr = findnext ()));
[AMA]  324:   showexit ();
                /* fertig */
[G34]  325: } /* end main() */
```

Listing. Der Quellcode zu »Move« kann mit jedem C-Compiler bearbeitet werden

DOS-Utilities
– nützliche Routinen für jeden Zweck

In diesem Artikel finden Sie nützliche Turbo Pascal-Utilities, die als eigenständige Programme oder als Module für andere Software sinnvoll eingesetzt werden können. Viele Probleme lassen sich so sehr einfach lösen.

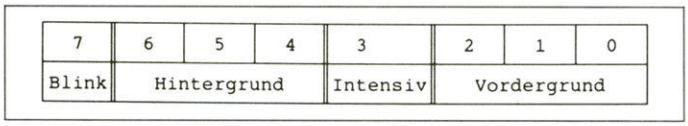


Bild 1. Dieses Byte steuert die Bildschirmattribute

Alle hier vorgestellten Programme haben einige Prozeduren gemeinsam. Um diese nicht jedesmal neu abtippen zu müssen, wurden die wichtigsten Routinen in Bibliotheken zusammengefaßt. Diese Prozeduren können Sie natürlich auch in anderen Turbo Pascal-Programmen verwenden. Bevor nun auf die einzelnen Utilities im Detail eingegangen wird, zunächst zu den Include-Dateien.

Die Include-Datei »Library«

»Library« ist eine für fast jedes Programm nützliche Toolbox, die eine schnellste Bildschirmausgaben erlaubt, sowie eine flexible Tastatureingabe (Listing 1). Drei Variablen spielen in dieser Toolbox eine wichtige Rolle:

Puffern ist eine typisierte boolsche Konstante. Ist sie auf »true« gesetzt, so wird beim Aufruf der Prozedur »Lies« die Tastatur gepuffert.

Screenaddr beinhaltet die Anfangsadresse des Bildschirms, so daß Ihre Programme unabhängig vom Bildschirmtyp laufen. Dies ist deshalb wichtig, weil die folgenden Ausgaberoutinen mit dieser Information arbeiten.

Attribut ist eine absolute Variable, mit der die Bildschirmattribute (Farben) für »Write(ln)« direkt angegeben werden können. Für Farbmonitore gilt die normale Farbpalette, die Turbo Pascal als typisierte Konstanten bereits liefert.

Für Monochrommonitore gilt folgende Zuordnung:

0 – keine Darstellung
1 – hell unterstrichen
9 – intensiv unterstrichen
10 – intensiv
32 – hell
112 – invers
120 – intensiv invers blinkend
129 – hell unterstrichen und blinkend
130 – hell blinkend
137 – intensiv unterstrichen und blinkend
240 – invers blinkend
248 – intensiv invers blinkend

Sind die drei Hintergrundbits gesetzt, so wird das Zeichen invers ausgegeben. Sind die drei Vordergrundbits gesetzt, so ist das Zeichen »normal«. Ist nur Bit 0 gesetzt, so wird das Zeichen unterstrichen. Bild 1 liefert die nötige Erklärung zu dieser Tabelle.

»Library.inc« enthält die Prozeduren »int24«, »int24s« und »int24r«.

int24s lenkt den Interrupt »24h« um.
int24r normalisiert Interrupt »24h« wieder.
int24 ist der eigentliche Interrupt-Handler.

Um ihn zu aktivieren, müssen Sie die globale Variable »di_reg« mit einem Fehlercode belegen (näheres siehe »int24f« in der Bibliothek »Librarye«). Es ist unbedingt erforderlich, vor Verlassen des Programms die Routine »int24r« aufzurufen!

Screen ermittelt die Bildschirmadresse und übergibt den Wert der globalen Variable »Screenaddr«. Diese Prozedur sollte stets zu Beginn eines Programms aufgerufen werden.

Cursor(anfang, ende) ändert die Größe des Cursors. Der Cursor besteht aus maximal 14 Mikrozeilen. In der Normaleinstellung hat der Parameter »anfang« den Wert 11 und der Parameter »ende« den Wert 12. Die Anweisung »Cursor(13, 2)« ergibt einen geteilten Cursor. »Cursor(32, 32)« bewirkt, daß der Cursor ausgeschaltet wird, da das fünfte Bit des Cursor-Startregisters zum Ein- und Ausschalten des Cursors dient. Versuchen Sie beim PC niemals, Bit 5 oder 6 zu setzen. Die unerwünschten Resultate können bis hin zum Absturz des Sytems führen.

CursorFrage(anfang, ende, seite) ist die Umkehrung von »Cursor« und dient zur Abfrage der Cursorform. Die gewünschte Bildschirmseite für die Abfrage wird dem Parameter »seite« übergeben (Standard »0«).

Adr(x, y) berechnet die Adresse für die Bildschirmposition »x,y« und liefert den Wert in der globalen Varibale »MemAdr«.

SetAdr(x, y) arbeitet wie »Adr«, nur wird die globale Variable »LastMem« auch belegt. Dies ist für die Prozedur »NextLine« äußerst wichtig.

XYGoto errechnet aus der gegenwärtigen Bildschirmadresse »MemAdr« die (x,y)-Position und setzt den Cursor an diese Stelle.

NextLine setzt die aktuelle Bildschirmkoordinate unter Verwendung von »succ(y)« auf die durch »SetAdr(x,y)« berechnete Position.

SchreibeZeichen(ch) schreibt das Zeichen »ch« mit dem aktuellen »Attribut« an die aktuelle Bildschirm-Koordinate (gesetzt durch »ADR« oder »SETADR«).

SchreibeSatz(st) arbeitet wie »SchreibeZeichen«, mit dem Unterschied, daß kein Zeichen, sondern ein ganzer String (»st«) geschrieben wird.

MoveChr(x, y, ch) »MoveChr« schreibt das Zeichen »ch« an die Stelle (x,y).

MoveStr(x, y, st) arbeitet wie »MoveChr«, mit dem Unterschied, daß kein Zeichen, sondern ein ganzer String (»st«) geschrieben wird.

MoveInt(n, i) schreibt die Integer-Variable »i« auf n Stellen formatiert an die aktuelle Bildschirm-Koordinate (gesetzt durch »ADR« oder »SETADR«).

MoveReal(nVor, nNach, r) schreibt die Real-Variable »r« formatiert auf »nVor« Vorkomma- und »nNach« Nachkommastellen an die aktuelle Bildschirm-Koordinate (gesetzt durch »ADR« oder »SETADR«).

Read2(ch) liest ein gepuffertes Zeichen (»ch«) von der Tastatur ein. Sie sollten beim erweiterten Tastaturcode nicht auf »(ch = #27) and keypressed«, sondern nur auf »ch = #0« abfragen!

Clear(ch) arbeitet wie »Read2«, nur wird der Tastaturpuffer vor dem Einlesen gelöscht (entspricht dem Befehl »Unitclear« beim UCSD-Pascal).

Lies(ch) richtet sich nach der globalen Variable »Puffern«, um zu entscheiden, ob ein Zeichen (»ch«) gepuffert eingelesen wird.
Warte(a, x, y, ch) schreibt die Aufforderung »Taste drücken« an die Bildschirmposition »x,y« (mit dem Attribut »a«). Der Tastaturcode der gedrückten Taste wird der Variablen »ch« übergeben (kein Test auf erweiterten Tastaturcode).
Signal sendet alle 50ms ein 1,5 kHz-Signal.
CtrlBreak(lesen, setzen) »CtrlBreak« setzt den Kontrollstatus für die Break-Tasten [Ctrl-C] und [Ctrl-Break] (erster Parameter). Dem zweiten Parameter wird der Status übergeben. Hier zwei Beispiele:
CtrlBreak(true,b) – Lies [Ctrl-C]-Status und gib Boolean in »b« aus.
CtrlBreak(false,b) – Belege [Ctrl-C]-Status mit Boolean »b«.
Bitte verwechseln Sie diese Routine nicht mit »cBreak« oder den Compileroptionen »C« und »U«!

```
[VB3]    0: (LIBRARY.INC)
[7PA]    1: ((C) A.H.Baqué,  Trierer Str. 64, 5300 Bonn
           1
[N3C]    2: (Erstellt für DOS international 7.1.88))
[000]    3:
[B6H]    4: const       int24Ofs : integer = 0;       {S
           icherung der Reg. ES,BX)
[K2A]    5:             int24Seg : integer = 0;       {d
           es INT24H)
[AMK]    6:               DI_REG : integer = 32767;   {S
           peicher für Fehlercode des INT24H)
[ZXD]    7:             Break24H : boolean = false;   {v
           on INT24F benutzt)
[5LE]    8:              Puffern : boolean = true;    {T
           astaur puffern)
[RNL]    9:       trueB : boolean = false;            {H
           ilfsvar. für <Ctrl,C>-Sicherung)
[7MA]   10: type           tregs = record              {R
           egister)
[S25]   11:                           case integer of
[AZD]   12:                            1 : ( ax,bx,cx,dx,
           bp,si,di,ds,es,flags : integer );
[BX9]   13:                            2 : ( al, ah, bl,
           bh, cl, ch, dl, dh : byte );
[TUO]   14:                         end;
[676]   15:             tstr255 = string[255];
[6LF]   16: var  Memadr,LastMem,                        {A
           kt. BIldschrimadresse)
[LPD]   17:           screenaddr : integer;             {B
           ildschrimadresse)
[8PG]   18:        Ca, Ce, wx, wy : byte;               {C
           ursorgröße/Cursorpos.}
[W9K]   19:             attribut : byte absolute dseg:8
           ; {Bildschirmattribut}
[Z7H]   20:       CtrlBreakStatus : boolean;           {C
           trl,C-Status sichern}
[FKH]   21:                     h : array[1..25,1..160]
           of byte; {Sicherung des Bildschirms}
[000]   22:
[000]   23:
[4MN]   24: procedure int24; {Interrupt-Handler, Fehler
           werden in DI_REG abgelegt}
[BH1]   25:   begin
[746]   26:    inline( $2E/$89/$3E/DI_REG/
[KM5]   27:            $b0/<3/$5d/$8b/$e5/$5d/$cf );
[TUO]   28:   end;
[000]   29:
[4U8]   30: procedure int24S; {Umlenken}
[Z65]   31:  var regs : tregs;
[BH1]   32:   begin
[375]   33:    with regs do begin
[7X1]   34:     ax := $3524;
[5C3]   35:     msdos( regs );
[TBA]   36:     int24Seg := es; {Für später retten}
[L94]   37:     int24Ofs := bx;
[7X1]   38:     ax := $2524;
[W82]   39:     ds := cseg;
[WD4]   40:     dx := ofs( int24 );
[5C3]   41:     msdos( regs );
[TUO]   42:    end;
[TUO]   43:   end;
[000]   44:
[PWA]   45: procedure int24R; {Zurücklenken}
[Z65]   46:  var regs : tregs;
[BH1]   47:   begin
[375]   48:    with regs do begin
[7X1]   49:     ax := $2524;
[214]   50:     ds := int24Seg;
[934]   51:     dx := int24Ofs;
```

```
[5C3]   52:     msdos( regs );
[TUO]   53:    end;
[TUO]   54:   end;
[000]   55:
[4RG]   56: procedure screen; {Bildschirmadresse ermitt
           eln}
[Z65]   57:  var regs : tregs;
[BH1]   58:   begin
[P04]   59:    regs.ah := $0F;
[D04]   60:    intr($10,regs);
[G85]   61:    if regs.al <> 7 then
[HY6]   62:     screenaddr := $b800 {Farbe}
[A11]   63:    else
[139]   64:     screenaddr := $b000; {Monochrom}
[TUO]   65:   end;
[000]   66:
[YCK]   67: procedure cursor( anfang, ende : byte ); {C
           ursor ändern}
[Z65]   68:  var regs : tregs;
[BH1]   69:   begin
[OA3]   70:    regs.ah := 1;
[L75]   71:    regs.ch := anfang;
[K25]   72:    regs.cl := ende;
[D04]   73:    intr($10,regs);
[TUO]   74:   end;
[000]   75:
[WHK]   76: procedure cursorfrage( seite : byte; var an
           fang, ende : byte );
[N48]   77:  {Cursorgröße abfragen}
[Z65]   78:  var regs : tregs;
[BH1]   79:   begin
[4X3]   80:    regs.ah := $03;
[X75]   81:    regs.bh := seite;
[D04]   82:    intr($10,regs);
[ZW4]   83:    anfang := regs.ch;
[HG4]   84:    ende := regs.cl;
[YW7]   85:    {dx : Position des Cursors }
[TUO]   86:   end;
[000]   87:
[3B7]   88: procedure adr( x, y : byte );
[B4G]   89:  {Adresse bei direkter Adressierung für die
           Pos. x,y}
[BH1]   90:   begin
[ETA]   91:    memadr := (pred(y)*5) shl 5 + pred(x) sh
           l 1;
[TUO]   92:   end;
[000]   93:
[P19]   94: procedure SetAdr( x, y : byte );
[DDK]   95:  {Wie Adr, nur wird Wert in LastMem zusätzl
           ich gesichert}
[BH1]   96:   begin
[9G2]   97:    adr(x,y);
[FB5]   98:    LastMem := Memadr;
[TUO]   99:   end;
[000]  100:
[6X5]  101: procedure xygoto;
[BH1]  102:   begin
[VBG]  103:    gotoxy(succ((memadr mod 160) shr 1), suc
           c(memadr div 160));
[YXN]  104:    {xpos von memadr +1, weil nächtes Zeiche
           n angesprochen werden soll ! }
[TUO]  105:   end;
[000]  106:
[86R]  107: procedure NextLine; {Geht in die nächste Ze
           ile, nur verwenden bei SetAdr}
[BH1]  108:   begin
[G56]  109:    memadr := LastMem + 160;
[F45]  110:    LastMem := MemAdr;
[TUO]  111:   end;
[000]  112:
[3RP]  113: procedure schreibezeichen( ch : char ); {Sc
           hreibt Zeichen an akt. Stelle}
[BH1]  114:   begin
[LBB]  115:    mem[ screenaddr : memadr ] := ord( ch );
[227]  116:    memadr := succ( memadr );
[3AC]  117:    mem[ screenaddr : memadr ] := attribut;
[227]  118:    memadr := succ( memadr );
[TUO]  119:   end;
[000]  120:
[3PP]  121: procedure SchreibeSatz( s : tstr255 ); {Sch
           reibt String an akt. Stelle}
[W63]  122:  var i : byte;
[BH1]  123:   begin
[SR8]  124:    for i := 1 to byte(s[0]) do begin
[L9C]  125:     mem[ screenaddr : memadr ] := ord( s[i]
           );
[227]  126:     memadr := succ( memadr );
[3AC]  127:     mem[ screenaddr : memadr ] := attribut;
[227]  128:     memadr := succ( memadr );
[TUO]  129:    end;
[TUO]  130:   end;
[000]  131:
[Z6N]  132: procedure moveChr( x, y : byte; ch : char )
           ; {Schreibt Zeichen an Pos. x,y}
[BH1]  133:   begin
[9G2]  134:    adr(x,y);
[396]  135:    SchreibeZeichen(ch);
```

```
[TUO]  136:    end;
[000]  137:
[76E]  138: procedure moveStr( x, y : byte; str255 : ts
           tr255 );
[ML9]  139:   {Schreibt String an Pos. x,y}
[BH1]  140:    begin
[9G2]  141:     adr(x,y);
[V96]  142:     SchreibeSatz(str255);
[TUO]  143:    end;
[000]  144:
[A4E]  145: procedure MoveInt( stellen : byte; i : inte
           ger );
[YN9]  146:   {Schreibt formatiert Integer}
[SY4]  147:   var s : string[9];
[BH1]  148:    begin
[245]  149:     str(i:stellen,s);
[815]  150:     SchreibeSatz(s);
[TUO]  151:    end;
[000]  152:
[T8D]  153: procedure MoveByte( stellen : byte; b : byt
           e );
[5D8]  154:   {Schreibt formatiert Byte}
[SY4]  155:   var s : string[9];
[BH1]  156:    begin
[Z25]  157:     str(b:stellen,s);
[815]  158:     SchreibeSatz(s);
[TUO]  159:    end;
[000]  160:
[RSK]  161: procedure MoveReal( stellenVor, stellenNach
           : byte; r : real );
[0C8]  162:   {Schreibt formatiert Real}
[555]  163:   var s : string[15];
[BH1]  164:    begin
[HAA]  165:     str(r:stellenVor:stellenNach,s);
[815]  166:     SchreibeSatz(s);
[TUO]  167:    end;
[000]  168:
[BW9]  169: procedure read2( var ch : char );
[5YC]  170:   {Liest gepuffert (!) Zeichen von Tastatur}
[Z65]  171:   var regs : tregs;
[BH1]  172:    begin
[SA3]  173:     regs.ah := 7;
[W93]  174:     msdos(regs);
[LH4]  175:     ch := chr(regs.al);
[TUO]  176:    end;
[000]  177:
[W9A]  178: procedure clear( var ch : char );
[Z6G]  179:   {Loscht Tastaturpuffer und liest dann Zeic
           hen ein}
[Z65]  180:   var regs : tregs;
[BH1]  181:    begin
[234]  182:     regs.ax := $0C07;
[W93]  183:     msdos(regs);
[LH4]  184:     ch := chr(regs.al);
[TUO]  185:    end;
[000]  186:
[959]  187: procedure Lies( var ch : char );
[Y1K]  188:   {Gepufferte Eingabe ? Richtet sich nach Pu
           ffern (global)}
[BH1]  189:    begin
[K95]  190:     if Puffern then
[R52]  191:      read2(ch)
[A11]  192:     else
[CL2]  193:      clear(ch);
[TUO]  194:    end;
[000]  195:
[YGE]  196: procedure warte( a, x, y : byte; var ch : c
           har );
[NH4]  197:   var ch1 : char;
[BH1]  198:    begin
[B84]  199:     attribut := a;
[7M8]  200:     MoveStr(x,y,'Taste drücken');
[D92]  201:     read2(ch);
[FD4]  202:     if ch = #0 then
[2E2]  203:      read2(ch1);
[TUO]  204:    end;
[000]  205:
[CP5]  206: procedure signal;
[BH1]  207:    begin
[UM2]  208:     sound(1400);
[NA2]  209:     delay(50);
[UZ2]  210:     nosound;
[TUO]  211:    end;
[000]  212:
[XGM]  213: procedure ctrlbreak( abfragen : boolean; va
           r erkennen : boolean );
[Z65]  214:   var regs : tregs;
[BH1]  215:    begin
[LX3]  216:     regs.ah := $33;
[R75]  217:     if abfragen then
[H13]  218:      regs.al := 0
[A11]  219:     else
[WA3]  220:      regs.al := 1;
[195]  221:     if erkennen then
[V43]  222:      regs.dx := 1
[A11]  223:     else
[YD3]  224:      regs.dx := 0;
[5C3]  225:     msdos( regs );
[KY6]  226:     erkennen := ( regs.dl = 1 );
[TUO]  227:    end;
[000]  228:
[ERH]  229: {Das Hauptprogramm für alle Programme, die
           die LIBRARY nutzen
[N3C]  230:   (Erstellt für DOS international 7.1.88)
[000]  231:
[W3C]  232:   begin   CBreak:=false bringt hier nichts
[FTA]  233:    ctrlbreak(true,ctrlbreakstatus);
[MP7]  234:    ctrlbreak(false,trueB);
[0B3]  235:    wx := wherex;
[6B3]  236:    wy := wherey;
[7C5]  237:    CursorFrage(0,Ca,Ce);
[DK2]  238:    screen;
[ZV9]  239:    move( mem[ screenaddr : 0 ], h, 4000 );
[WZ1]  240:    int24S;
[YX6]  241: ----  Procedure-Aufruf ----
[PZ1]  242:    int24R;
[L0A]  243:    move( h, mem[ screenaddr : 0 ], 4000 );
[B74]  244:    gotoxy(wx,wy);
[3W3]  245:    Cursor(Ca,Ce);
[99B]  246:    CtrlBreak(false,CtrlBreakStatus);
[PC1]  247:   end.)
```

Listing 1. Quellcode der Include-Datei »library.inc«

»LibraryE« – Erweiterungen für die Toolbox »Library«

Folgende Prozeduren (Listing 2) stellen eine Ergänzung zur Bibliothek »Library« dar und sind nur gemeinsam mit dieser Toolbox und der Bibliothek »window.inc« lauffähig. Diese Routinen ersparen Ihnen eine Menge Ärger bei Zugriffen auf externe Datenträger:

Sicherung PC-Besitzer mit nur einem Laufwerk kennen wahrscheinlich die Meldung »Diskette in Laufwerk x: legen und Taste drücken«. Nun ist es aber nicht besonders schön, wenn diese Meldung während des Programmablaufs am Bildschirm erscheint und Ihren Bildschirmaufbau durcheinander bringt. Die Routine »Sicherung« stellt für diese Systemmeldung ein Window zur Verfügung, in welchem die Meldung abgefangen wird.

Entsicherung Das durch »Sicherung« geöffnete Window wird gelöscht und Ihr Bildschirm sieht genauso aus wie vor dem Aufruf der Routine »Sicherung«.

int24F Sollte ein kritischer Fehler aufgetreten sein (vorausgesetzt »int24S« wurde vorher aufgerufen), so zeigt diese Routine die Fehlermeldung in einem Window an. Dieses wird aus der globalen Konstanten »di_reg« ermittelt. Zusätzlich wird die boolsche Variable »break24h« mit »true« belegt.

```
[5U3]    0: {LIBRARYE.INC}
[06C]    1: {(Erstellt für DOS international 7.1.88)}
[000]    2:
[VE6]    3: type Tstr80 = string[80];
[N5A]    4: {DOS-MELDUNG : DISKETTE IN LAUFWERK X: LEGE
            N..
[GNK]    5:     Bei festplattenbenutzern tritt diese Mel
            dung schon mal auf !
[X27]    6:     Wer dies also braucht :}
[RNB]    7: var sicherhilf : array[1..4,1..90] of byte;
[000]    8:
[7F6]    9: procedure sicherung;
[BH1]   10:    begin
[X7B]   11:     MakeWindow(1,32,32,32,1,4,45,4,SicherHil
            f[1,1]);
[Y43]   12:     gotoxy(1,4);
[TUO]   13:    end;
[000]   14:
[V98]   15: procedure Entsicherung;
[BH1]   16:    begin
[FUA]   17:     RestoreWindow(1,4,45,4,SicherHilf[1,1]);
[TUO]   18:    end;
[000]   19:
[005]   20: procedure Int24F;
```

```
[M1E]   21:  (Abfrage, ob schwerwiegender Fehler auftra
          t)
[N6B]   22:  const s : array[1..13] of string[13] = (
[3C5]   23:         'Schreibschutz',
[2D5]   24:         'Geräteeinheit',
[VA4]   25:         'Nicht bereit ',
[D34]   26:         'Statement ?  ',
[1D4]   27:         'Datenfehler  ',
[PN4]   28:         'Aufruflänge ?',
[434]   29:         'Suchfehler   ',
[Y24]   30:         'Gerätart    ?',
[UU4]   31:         'Sektor n.gef.',
[944]   32:         'Papierende   ',
[V65]   33:         'Schreibfehler',
[R24]   34:         'Lesefehler   ',
[7Z3]   35:         'Allg. Fehler ');
[9A8]   36:  var h : array[1..3,1..60] of byte;
[PF2]   37:      a : byte;
[D92]   38:     ch : char;
[BH1]   39:   begin
[6L4]   40:    Break24H := false;
[ML7]   41:    if DI_Reg <> 32767 then begin
[HN4]   42:     Break24H := true;
[WH2]   43:     signal;
[T14]   44:     a := attribut;
[GN7]   45:     MakeWindow(2,112,15,32,10,12,30,3,H);
[MC4]   46:     attribut := 15;
[7ZA]   47:     MoveStr(11,12,'Kritischer Fehler '#19);
[X94]   48:     attribut := 112;
[FY7]   49:     if (DI_REG > 13) or (DI_REG < 1) then
[1H2]   50:      DI_REG := 13;
[187]   51:     MoveStr(11,13,s[DI_Reg]);
[WT3]   52:     warte(15,11,14,ch);
[GZ6]   53:     RestoreWindow(10,12,30,3,H);
[MH3]   54:     DI_Reg := 32767;
[B84]   55:     attribut := a;
[TU0]   56:    end;
[TU0]   57:   end;
[000]   58:
```

Listing 2. »librarye.inc« ist eine Erweiterung zur Toolbox »library.inc«

Schnelle Fensterverwaltung mit der Window-Toolbox

Viele Anwendungen lassen sich durch Fenstertechnik besser verwirklichen. Andere Probleme sind ohne diese Fenster erst gar nicht zu lösen. Die Fenster-Routinen dieser Include-Datei (Listing 3) sind sehr vielseitig, sehr schnell und optisch äußerst variabel. Dies wurde durch direkte Adressierung realisiert. Für diese Window-Technik benötigen Sie die Bibliothek »Library.inc«. Der Aufbau eines 80x25 Zeichen großen Fensters (ganzer Bildschirm) und der Restaurierung des alten Bildschirms benötigt auf einem PC (8088/4,77MHz) nur 1/100 Sekunde. Zwei Prozeduren existieren hierfür:

```
MakeWindow( art, a, a1, fuell, x, y, anzx, anzy : byte; var addr )
RestoreWindow( x, y, anzx, anzy : byte; var addr )
```

Der Parameter »art« erlaubt eine flexible Gestaltung des Windows. Zehn Rahmen stehen dem Benutzer zur Verfügung (Bild 2). Der Parameter »a« bestimmt das Attribut, mit dem das Window gefüllt wird. »a1« bestimmt das Attribut, mit dem der Rahmen gezeichnet wird. »fuell« ist das Zeichen (ASCII/EBCDIC), mit dem das Window gefüllt wird. »x,y« sind die Koordinaten der linken oberen Ecke. »anzx,anzy« geben die Anzahl der Zeichen in x/y-Richtung (ab der Position »x,y«) an. »Addr« ist die Adresse eines für das Window bereitgestellten Speicherbereichs, in dem der Bildschirm gespeichert wird. So können ganz nach Geschmack statische und dynamische Variablen (auch verkettete Listen sind denkbar) übergeben werden. Das kleine Demo-Programm »win_demo.pas« (Listing 4) zeigt Ihnen die hohe Geschwindigkeit und Vielseitigkeit dieser Toolbox.

```
[D33]    0: {WINDOW.INC}
[26C]    1: {(Geändert für DOS international 7.1.88)}
[000]    2:
[2GN]    3: procedure MakeWindow( art, a, a1, fuell, x,
           y, anzx, anzy : byte; var addr );
[3Z9]    4:  var c, c0,  count1, c1, o, c2 : integer;
[XN3]    5:              LO, RO, WO, LU, RU,
[895]    6:                WU, S, SR, SL, WM : byte;
[000]    7:
[T18]    8:  procedure best( art : byte );
[BH1]    9:    begin
[623]   10:     case art of
[840]   11:       0,
[WC2]   12:       1 : begin;
[S08]   13:             LO := 218;                RO := 191
                      ;   LU := 192;
[848]   14:             RU := 217;                 S := 179
                      ;   WO := 196;
[XH1]   15:             WU := 196;
[E96]   16:             if art = 0 then begin
[SR8]   17:              WM := 196;               SL := 195
                      ;   SR := 180;
[TU0]   18:             end;
[TU0]   19:           end;
[A40]   20:       2,
[C92]   21:       7 : begin
[BY7]   22:             LO := 201;                RO := 187
                      ;   LU := 200;
[H28]   23:             RU := 188;                 S := 186
                      ;   WO := 205;
[PP1]   24:             WU := WO;
[W96]   25:             if art = 7 then begin
[SR8]   26:              SL := 199;               SR := 182
                      ;   WM := 196;
[TU0]   27:             end;
[TU0]   28:           end;
[H40]   29:       9,
[B40]   30:       3,
[B92]   31:       6 : begin
[X18]   32:             LO := 213;                RO := 184
                      ;   WO := 205;
[WP7]   33:             LU := 192;                RU := 217
                      ;    S := 179;
[XH1]   34:             WU := 196;
[U96]   35:             if art = 6 then begin
[SR8]   36:              WM := 196;               SL := 195
                      ;   SR := 180;
[TU0]   37:             end;
[OA6]   38:             if art = 9 then begin
[7R8]   39:              WM := 205;               SR := 181
                      ;   SL := 198;
[TU0]   40:             end;
[TU0]   41:           end;
[C40]   42:       4,
[D92]   43:       8 : begin
[VY7]   44:             LO := 214;                RO := 183
                      ;   LU := 211;
[A48]   45:             RU := 189;                 S := 186
                      ;   WO := 196;
[PP1]   46:             WU := WO;
[Y96]   47:             if art = 8 then begin
[AH1]   48:              SL := 199;
[FH1]   49:              SR := 182;
[6P1]   50:              WM := WO;
[TU0]   51:             end;
[TU0]   52:           end;
[A92]   53:       5 : begin
[4M8]   54:             LO := 219;                RO := LO;
                        LU := LO;
[CF8]   55:             RU := LO;                  S := LO;
                        WU := 220;
[NH1]   56:             WO := 223;
[TU0]   57:           end;
[TU0]   58:     end;
[TU0]   59:    end;
[000]   60:
[PVE]   61:   procedure h( a : integer; art : byte; t :
          integer );
[8Z7]   62:     var ch, art1, count : integer;
[BH1]   63:      begin
[B34]   64:       art1 := a + art;
[A33]   65:       count := t+2;
[C82]   66:       ch := t+c0;
[X52]   67:       repeat
[D2U]   68:        memW[screenaddr:count] := art1; {Eine
          n Hauch schneller, als einzeln zuzuweisen}
[ZT4]   69:        count := count+2;
[GF4]   70:       until count = ch;
[TU0]   71:      end;
[000]   72:
[BH1]   73:   begin
[OT2]   74:    best(art);
[864]   75:    o := ofs(addr);
[9G2]   76:    adr(x,y);
[HO4]   77:    c := anzx shl 1;
[GE6]   78:    c0 := pred(anzx) shl 1;
[2H3]   79:    c2 := a1 shl 8;
[RXD]   80:    Move( mem[screenaddr:memadr], addr, anzx
          shl 1 );
```

```
[Y5A]   81:     memW[screenaddr:memadr] := c2 + L0;
[TB4]   82:     h(c2,W0,memadr);
[11B]   83:     memW[screenaddr:memadr+c0] := c2 + R0;
[LS5]   84:     memadr := memadr+160;
[202]   85:     o := o + c;
[8KG]   86:     Move( mem[ screenaddr : memadr ], mem[ s
          eg( addr ) : o ] , c );
[323]   87:     c1 := memadr;
[AX6]   88:     h(a shl 8,FUELL,memadr);
[RRT]   89:     memW[screenaddr:memadr] := c2 + S; {Eine
          n Hauch schneller, als einzeln zuzuweisen}
[54H]   90:     memW[screenaddr:memadr+c0] := memW[scree
          naddr:memadr];
[KG9]   91:     for count1 := 1 to anzy-3 do begin
[LS5]   92:      memadr := memadr+160;
[202]   93:      o := o + c;
[8KG]   94:      Move( mem[ screenaddr : memadr ], mem[
          seg( addr ) : o ] , c );
[ZNF]   95:      Move(mem[screenaddr:c1],mem[screenaddr:
          memadr],c);
[TU0]   96:     end;
[202]   97:     o := o + c;
[LS5]   98:     memadr := memadr+160;
[WRE]   99:     Move( mem[screenaddr:memadr], mem[seg(ad
          dr):o] , c);
[A6A]  100:     memW[screenaddr:memadr] := c2 + LU;
[4T5]  101:     if art in [3,5,6,9,10] then
[3D4]  102:      h(c2,WU,memadr)
[A11]  103:     else
[EVL]  104:      move(mem[screenaddr:c1-158],mem[screena
          ddr:memadr+2],(anzx-2) shl 1);
[Z1B]  105:     memW[screenaddr:memadr+c0] := c2 + RU;
[B36]  106:     if art in [0,6..10] then
[9L7]  107:      if anzy >= 5 then begin
[YW3]  108:       memadr := c1+160;
[Z5A]  109:       memW[screenaddr:memadr] := c2 + SL;
[C06]  110:       if art in [6,7,9,10] then
[BB4]  111:        h(c2,WM,memadr)
[A11]  112:       else
[EVL]  113:        move(mem[screenaddr:c1-158],mem[scree
          naddr:memadr+2],(anzx-2) shl 1);
[M1B]  114:       memW[screenaddr:memadr+c0] := c2 + SR;
[TU0]  115:      end;
[TU0]  116:    end;
[000]  117:
[S5K]  118: procedure RestoreWindow( x, y, anzx, anzy :
          byte; var addr );
[9X6]  119:  var  count, o, w : integer;
[BH1]  120:   begin
[864]  121:    o := ofs(addr);
[514]  122:    w := anzx shl 1;
[9G2]  123:    adr(x,y);
[S29]  124:    for count := 1 to anzy do begin
[S2G]  125:     Move( mem[ seg( addr ) : o] , mem[ scre
          enaddr : memadr ], w );
[LS5]  126:      memadr := memadr+160;
[A12]  127:      o := o + w;
[TU0]  128:     end;
[TU0]  129:    end;
```

Listing 3. Zur schnellen Fensterverwaltung dient die Bibliothek »window.inc«

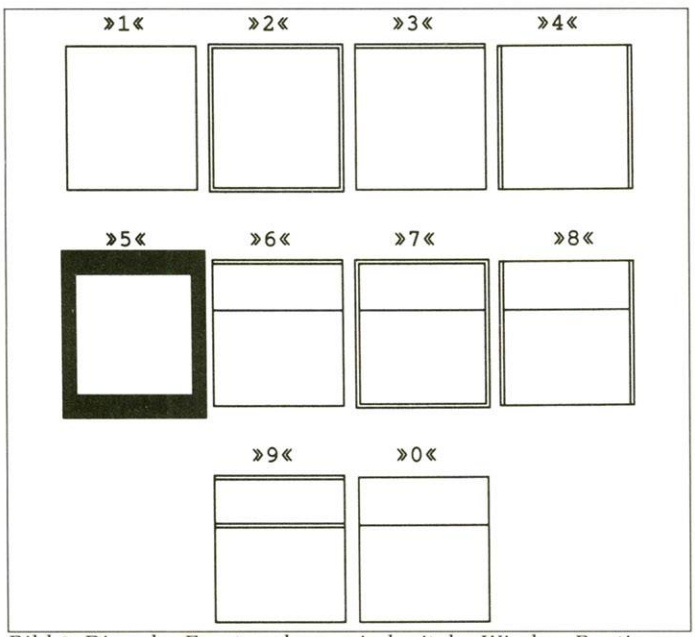


Bild 2. Die zehn Fensterrahmen sind mit der Window-Routine zu realisieren

```
[66C]    0:   {WINDOWDEMO für DOS international 10.01.8
           8}
[000]    1:
[C55]    2:   {$I-,R-,V-,C-,U-,D-,K-}
[WP3]    3:   {$I LIBRARY.INC}
[R64]    4:   {$I WINDOW.INC}
[000]    5:
[F94]    6:   var a, i : byte;
[D92]    7:         ch : char;
[BH1]    8:   begin
[DK2]    9:    screen;
[L86]   10:    for i := 0 to 9 do begin
[41L]   11:     if i = 5 then  {Ränder sind bei Attribu
          t 112 hier nicht sichtbar}
[WB1]   12:      a := 15
[A11]   13:     else
[VC1]   14:      a := 112;
[BP7]   15:     MakeWindow(i,15,a,177, 1,1,80,25,h);
[MC4]   16:     attribut := 15;
[UEF]   17:     MoveStr(22,2,'Schnelle und flexible Win
          dow-Technik');
[X94]   18:     attribut := 112;
[CW1]   19:     Adr(40,12);
[E24]   20:     MoveByte(1,i);
[LT3]   21:     Warte(248,2,25,ch);
[XV6]   22:     RestoreWindow(1,1,80,25,h);
[TU0]   23:    end;
[5T0]   24:   end.
```

Listing 4. Dieses Pascal-Programm demonstriert die Leistungsfähigkeit der Fenster-Routine

»Tools« enthält Routinen, die Sie nicht so oft brauchen

Die Routinen aus Listing 5 sind gegen alle nur denkbaren Fehlerbedingungen, im Gegensatz zu den Äquivalenten in Turbo Pascal (sofern diese überhaupt existieren), immun.

AktDrive(drive) Nach Aufruf dieser Prozedur enthält der Parameter »drive« die Nummer des aktuellen Laufwerks (A = 0,B = 1,C = 2,D = 3...)

NewDrive(drive) versucht »drive« zum aktuellen Laufwerk zu machen und gibt wie »AktDrive« die Nummer des aktuellen Laufwerks in »drive« zurück. Auf diese Weise können Sie testen, ob Ihr Versuch zum Erfolg geführt hat, oder ob das »alte« Laufwerk beibehalten wurde.

Speicher(SektorenProCluster, FreieCluster, ByteProSektor, GesamtCluster : integer; Drive : byte) ermittelt den freien Speicher von externen Datenträgern, wobei die Kapazität gleich »SektorenProCluster« mal »GesamtCluster« mal »ByteSektor« ist. Der »freie Speicher« ist gleich »FreieCluster« mal »SektorenProCluster« mal »ByteJeSektor«. »Drive« gibt das gewünschte Laufwerk an (A = 65,B = 66,C = 67...).

Rollen1 schiebt den Bildschirm sehr schnell (schneller als die entsprechende BIOS-Routine) um eine Zeile nach oben.

Fehlb(c, msg) gibt eine Fehlermeldung, die in dem String »msg« enthalten sein muß, in einem Fenster aus und stellt dem Benutzer die magische Frage »A>bbruch W>iederholen I>gnorieren«. Die Antwort wird nach »c« übergeben.

Print(ch, BusyB) versucht ein Zeichen auf den parallelen Drucker (LPT-Nummer in typisierter Konstante »LPTnr«) auszugeben. »BusyB« liefert das Ergebnis dieses Versuchs. Ist nämlich »BusyB« gleich Null, so ging alles gut, ansonsten enthält diese Variable einen Fehlercode, der von der Prozedur »PrintMsg« ausgewertet werden kann. Zu bemerken ist vor allem, daß Ihr Programm nicht mehr abstürzen kann, was bei »Write(LST,...)« durchaus wahrscheinlich ist. Folgende Fehlerbedingungen lassen sich durch »BusyB« feststellen (siehe »PrintMsg«) :

Bit 7 gesetzt – Drucker beschäftigt
Bit 5 gesetzt – Papierende
Bit 3 gesetzt – Drucker nicht eingeschaltet
Bit 1 gesetzt – Zeitüberschreitung

PrintMsg(code) wertet den Fehlercode »code« der Prozedur »Print« aus und setzt ihn in einen String um. Das Ergebnis dieser Funktion ist also besagte Meldung.
Print1(ch, f) rufen Sie auf, um ein Zeichen an den Drucker zu senden, wobei die Variable »ch« das zu druckende Zeichen als Byte enthält. Die boolsche Variable »f« enthält anschließend das Ergebnis dieses Druckversuchs (»true« : Fehler aufgetreten). Intern funktioniert die Prozedur folgendermaßen: »Print1« versucht über die Prozedur »Print« ein Zeichen auszugeben. Schlägt dies fehl, so wird mit der Prozedur »FehlB« der weitere Verlauf bestimmt. Es können somit keine Daten verloren gehen.

```
[5H2]    0: {TOOLS}
[000]    1:
[U85]    2: CONST LPTnr : byte = 1;
[000]    3:
[A7C]    4: procedure aktdrive( var drive : byte );
[Z65]    5:  var regs : tregs;
[BH1]    6:   begin
[DY3]    7:    regs.ah := $19;
[W93]    8:    msdos(regs);
[2N6]    9:    drive := succ(regs.al);
[TU0]   10:   end;
[000]   11:
[88C]   12: procedure newdrive( var drive : byte );
[SGB]   13:  { Andert das aktuelle Laufwerk        }
[ZU4]   14:  var  aqt : byte;
[X33]   15:       regs : tregs;
[HC2]   16:         ch : byte;
[BH1]   17:  begin
[2G3]   18:   aqt := drive - 65;
[G04]   19:   regs.ah := $0E;
[HW4]   20:   regs.dl := aqt;
[W93]   21:   msdos(regs);
[UL2]   22:   ch := aqt+65;
[PK8]   23:   aktdrive(ch);    { Kontrolle }
[MX2]   24:   drive := ch;
[TU0]   25:  end;
[000]   26:
[0NH]   27: procedure aktdir( var pfadname : tstr255; d
          rive : byte);
[Z65]   28:  var regs : tregs;
[BH1]   29:   begin
[375]   30:    with regs do begin
[LG1]   31:     ah := $47;
[TN2]   32:     dx := drive;
[PF6]   33:     ds := seg(pfadname[1]);
[1F6]   34:     si := ofs(pfadname[1]);
[TU0]   35:    end;
[W93]   36:    msdos(regs);
[EA2]   37:  {  INT24F;}
[1L4]   38:    pfadname[0] := #80;
[8NB]   39:    pfadname[0] := chr(pred(pos(#0,pfadname)
        ));
[TU0]   40:   end;
[000]   41:
[5XS]   42: procedure speicher( var sektorenProCluster,
        freieCluster, byteProSektor,
[G5C]   43:                     gesamtCluster : integer
        ; drive : byte );
[Z65]   44:  var regs : tregs;
[BH1]   45:   begin
[5Y3]   46:    regs.ah := $36;
[W2L]   47:    regs.dx := drive - 64;        {physische
        Laufwerksnummer (byte)}
[5C3]   48:    msdos( regs );
[D13]   49: {   Int24F;}
[1C8]   50:    if regs.ax = $FFFF then begin
[6A7]   51:     sektorenProCluster := 0;
[VT4]   52:     freieCluster := 0;
[XC5]   53:     byteProSektor := 0;
[DLO]   54:    end
[YS2]   55:    else begin
[9H9]   56:     sektorenProCluster := regs.ax;
[CA7]   57:     freieCluster := regs.bx;
[YL7]   58:     byteProSektor := regs.cx;
[TU0]   59:    end;
[GD7]   60:    gesamtCluster := regs.dx;
[TU0]   61:   end;
[000]   62:
[PV5]   63: procedure rollen1;
[BH1]   64:   begin
[BBE]   65:    move(mem[screenaddr:160],mem[screenaddr:
        0],3840);
[LZA]   66:    fillchar(mem[screenaddr:3840],160,32);
[TU0]   67:   end;
[000]   68:
[4LC]   69: procedure FehlB( var c : byte; msg : tstr80
        );
[7GB]   70:  const   x = 10; y = 10; xs = 39; xss = 78;
        ys = 4; a = 7;
[YS6]   71:        st0 = 'Fehlerbedingung !';
[Y8C]   72:        st1 = 'A>bbruch  W>iederholen  I>gno
        rieren';
[WCB]   73:  var     h : array[1..ys,1..xss] of byte;
[D92]   74:         ch : char;
[BH1]   75:   begin
[559]   76:    MakeWindow(1,a,a,32,x,y,xs,ys,h);
[B84]   77:    attribut := a;
[UA4]   78:    SetAdr(succ(x),y);
[529]   79:    SchreibeSatz(st0);   NextLine;
[U09]   80:    SchreibeSatz(msg);   NextLine;
[MP5]   81:    SchreibeSatz(st1);
[X52]   82:    repeat
[D92]   83:     read2(ch);
[FD4]   84:     if ch = #0 then
[R52]   85:      read2(ch)
[A11]   86:     else
[YE4]   87:      ch := upcase(ch);
[7T5]   88:    until (ch in ['A','W','I']);
[E43]   89:    c := ord(ch);
[YP8]   90:    RestoreWindow(x,y,xs,ys,h);
[TU0]   91:   end;
[000]   92:
[A1E]   93: procedure print( ch : byte; var busyB : byt
        e );
[Z65]   94:  var regs : tregs;
[BH1]   95:   begin
[KW3]   96:    regs.ah := $00;
[734]   97:    regs.al := ch;
[SA7]   98:    regs.dx := pred(LPTnr);
[D04]   99:    intr($17,regs);
[6Z8]  100:    busyB := regs.ah and 41; {Bit 5,3,1}
[TU0]  101:   end;
[000]  102:
[A8D]  103: function PrintMsg( code : byte ) : tstr80;
[BH1]  104:   begin
[GD7]  105:    if boolean(code and 8) then
[EOD]  106:     Printmsg := 'Drucker nicht eingeschalte
        t'
[53A]  107:    else if boolean(code and 1) then
[HUA]  108:     Printmsg := 'Zeituberschreitung'
[FY9]  109:    else if boolean(code and 32) then
[HM7]  110:     Printmsg := 'Papierende'
[A11]  111:    else
[HFA]  112:     PrintMsg := 'Unbekannter Fehler';
[TU0]  113:   end;
[000]  114:
[WHD]  115: procedure print1( ch : byte; var f : boolea
        n );
[T94]  116:  var c, b : byte;
[BH1]  117:   begin
[743]  118:    print(ch,b);
[BO3]  119:    if b > 0 then
[TC2]  120:     f := true;
[ER1]  121:    INT24F;
[NA6]  122:    if BREAK24H then begin
[654]  123:     BREAK24H := false;
[TC2]  124:     f := true;
[TU0]  125:    end;
[025]  126:    if f then begin
[R26]  127:     fehlB(c,Printmsg(b));
[A22]  128:     case c of
[3M1]  129:      {65:;}
[784]  130:      87 : print1(ch,f);
[D44]  131:      73 : f := false;
[TU0]  132:     end;
[TU0]  133:    end;
[TU0]  134:   end;
[000]  135:
```

Listing 5. »Tools« enthält diverse Turbo Pascal-Routinen

Die Include-Datei »Readstr.inc« dient ausschließlich zur Eingabe von Zeichenketten

Die Prozedur »ReadStr« ist eine sehr komfortable und vielseitige Routine, um Zeichenketten (Strings) einzulesen (Listing 6). »ReadStr« benutzt folgende 6 Parameter:
str255 – der einzulesende Stringbereich – enthält alle Zeichen, die in dem String vorkommen dürfen
anzahl – die Anzahl der einzulesenden Zeichen
x1,y1 – die Koordinaten für den Einleseprozeß
nurgross – ist dieser Parameter »true«, dann wird alles in Großbuchstaben umgewandelt (auch Umlaute).
Beim Editieren stehen verschiedene Tasten zur Verfügung, deren Funktionen dem Benutzer in einem Window erklärt werden, so daß es sich erübrigt, hier näher darauf einzugehen. Die Routine ändert den Cursor automatisch, setzt ihn aber beim Verlassen der Prozedur wieder auf die

ursprüngliche Größe. Die Variable »str255« muß vor dem Aufruf von »ReadStr« initialisiert werden. Wird die Taste [Esc] gedrückt, so bricht die Routine ab und liefert im Längenbyte von »str255« den Wert 255. Die Testanweisung, ob [Esc] gedrückt wurde, sieht dann wie folgt aus:

```
if str255[0] = #255 then ...
```

Bevor Sie diese Routine aufrufen, fügen Sie unbedingt folgende Anweisung ein:

```
Type TSetOfChar = Set of Char;
```

Diese Routine wandelt auch Umlaute in Klein- beziehungsweise Großbuchstaben um (im Gegensatz zur Anweisung »upcase« von Turbo Pascal) und läßt sich beliebig erweitern.
Bei dem Beispiel

```
ReadStr(s,[#4..#19,#32..#64,#128..#255],12,40,10,true);
```

wird ein String »s« ab der Position »40,10« gelesen. »ReadStr« liest maximal zwölf Zeichen und wandelt alle in Großbuchstaben um (»true«).Gültige Zeichen, also Zeichen, die beim Einlesen zugelassen werden, sind nur jene, die in den eckigen Klammern (»Set«) stehen: »#4..#19, #32..#64 und #128..#255«.

```
[BD3]    0: {READSTR.INC}
[26C]    1: {(Geandert für DOS international 7.1.88)}
[000]    2:
[7LK]    3:  procedure readstr( var str255 : tstr255; b
           ereich : tsetofchar;
[DEC]    4:                      anzahl, x1, y1 : byte;
           Nurgross : boolean );
[PXR]    5:  {str255 muß vorher initialisiert werden, e
           ntweder mit einem String, oder
[VO7]    6:  halt als Leer-String !}
[EKU]    7:   const  ERKL1 : string[68] = '<ESC> Abbruc
           h  Edit.: <'#27','#26'> Bewegen, <Einfg/Ins>
             Insert/Overwrite,';
[FMR]    8:           ERKL2 : string[60] = '<Entf/Del> U
           nterm Cursor weg, <Rück./Back.> Vorm Cursor
            weg,';
[XTP]    9:           ERKL3 : string[62] = '<F1> Alles w
           eg, <F2> Ab Cursor löschen, <F3> Vor Cusor l
            oschen';
[F6V]   10:           ERKL4 : string[67] = 'Wandeln Groß
           -Kleinbuchstaben Tasten <Ctrl> groß oder  <A
            lt> klein +';
[S2H]   11:           ERKL5 : string[41] = '<F1> Alles <
           F2> Vor Cursor <F3> Ab Cursor';
[CV2]   12:   var  ca, ce,
[SO4]   13:          i, pos : byte;
[D92]   14:              ch : char;
[LM4]   15:        Cstr255 : tstr255;
[5D5]   16:      raus, ins : boolean;
[AV7]   17:           hilf : array[1..9,1..140] of byte
           ;
[000]   18:
[WKG]   19:   procedure loeschChr( var st : tstr255; an
           f, lanz : byte );
[BH1]   20:     begin
[3Y6]   21:      delete( str255, anf, lanz );
[VO5]   22:      if lanz = 1 then
[DO9]   23:       MoveChr(x1+length(str255),y1,#32)
[A11]   24:      else
[147]   25:       MoveStr( x1, y1, Cstr255 );
[AT6]   26:      MoveStr( x1, y1, str255 );
[4D7]   27:      gotoxy( x1 + pred(anf), y1 );
[TUO]   28:     end;
[000]   29:
[CTF]   30:   procedure wandle( anf, anz : byte; gross
           : boolean );
[W63]   31:    var i : byte;
[BH1]   32:     begin
[VA6]   33:      for i := anf to anz do
[494]   34:       if gross then
[U28]   35:        if str255[ i ] in ['a'..'z'] then
[MM8]   36:         str255[ i ] := chr(ord(str255[i])-3
           2)
[A11]   37:        else
[XU4]   38:         case str255[ i ] of
[OF4]   39:           #132 : str255[ i ] := #142;
[YF4]   40:           #148 : str255[ i ] := #153;
[6EC]   41:           #129 : str255[ i ] := #154; {Beli
           ebig zu erweitern}
[DLO]   42:         end
[A11]   43:        else
[378]   44:         if str255[i ] in ['A'..'Z'] then
[YT9]   45:          str255[ i ] := chr( ord( str255[ i
           ] ) + 32 )
[A11]   46:         else
[XU4]   47:          case str255[ i ] of
[2F4]   48:           #142 : str255[ i ] := #132;
[PF4]   49:           #153 : str255[ i ] := #148;
[2EC]   50:           #154 : str255[ i ] := #129; {Beli
           ebig zu erweitern}
[TUO]   51:         end;
[AU6]   52:      moveStr( x1, y1, str255 );
[TUO]   53:     end;
[000]   54:
[KM3]   55:   procedure p;
[BH1]   56:     begin
[UN4]   57:      i := x1 + pred(pos);
[995]   58:      moveChr( i, y1, ch );
[6V5]   59:      gotoxy( succ(i), y1 );
[784]   60:      pos := succ( pos );
[TUO]   61:     end;
[000]   62:
[BH1]   63:    begin
[124]   64:     i := attribut;
[7X8]   65:     MakeWindow(7,112,7,32, 5,17,70,9,hilf);
[7N5]   66:     CursorFrage(0,ca,ce);
[G93]   67:     Cursor(1,13);
[X94]   68:     attribut := 112;
[PPC]   69:     MoveStr(28,18,'Editierbare Zeichenkette
           ');
[013]   70:     SetAdr(6,20);
[8K5]   71:     SchreibeSatz(ERKL1);
[ZV2]   72:     NextLine;
[AK5]   73:     SchreibeSatz(ERKL2);
[ZV2]   74:     NextLine;
[CK5]   75:     SchreibeSatz(ERKL3);
[ZV2]   76:     NextLine;
[EK5]   77:     SchreibeSatz(ERKL4);
[ZV2]   78:     NextLine;
[GK5]   79:     SchreibeSatz(ERKL5);
[L94]   80:     attribut := i;
[5G2]   81:     ins := true;
[TW3]   82:     raus := false;
[KT6]   83:     Cstr255[0] := chr( anzahl );
[P7A]   84:     fillchar( Cstr255[1], anzahl, chr(32) )
           ;
[157]   85:     moveStr( x1, y1, Cstr255 );
[AR6]   86:     MoveStr( x1, y1, Str255 );
[VX6]   87:     pos := succ(Length(Str255));
[BL7]   88:     gotoxy( x1+pred(pos), y1 );
[X52]   89:     repeat
[6L2]   90:      lies(ch);
[S95]   91:      if Nurgross then
[H84]   92:       ch := upcase( ch );
[UY5]   93:      if ch = #0 then begin
[862]   94:       read2( ch );
[RS2]   95:       case ch of
[R52]   96:         #71 : begin
[Z84]   97:                gotoxy( x1, y1 );
[KP1]   98:                pos := 1;
[TUO]   99:               end;
[F62]  100:         #79 : begin
[679]  101:                gotoxy( x1 + length(str255),
           y1 );
[V67]  102:                pos := succ(length(str255));
[TUO]  103:               end;
[B62]  104:         #59 : begin
[NA2]  105:                str255 := '';
[157]  106:                moveStr( x1, y1, Cstr255 );
[KP1]  107:                pos := 1;
[Z84]  108:                gotoxy( x1, y1 );
[TUO]  109:               end;
[GAF]  110:         #60 : loeschChr( str255, pos, lengt
           h(str255)-pred(pos) );
[N52]  111:         #61 : begin
[C17]  112:                loeschChr( str255  1, pos );
[KP1]  113:                pos := 1;
[AW3]  114:                gotoxy(x1,y1);
[TUO]  115:               end;
[207]  116:         #94 : if not Nurgross then
[SD9]  117:                Wandle(1, length(str255), tr
           ue);
[507]  118:         #95 : if not Nurgross then
[UB5]  119:                Wandle(1, pos, true);
[807]  120:         #96 : if not Nurgross then
[XV9]  121:                Wandle(pos, length(str255),
           true);
[KH6]  122:        #104 : if not Nurgross then
[GU9]  123:                Wandle(1, length(str255), fa
           lse);
[PH6]  124:        #105 : if not Nurgross then
[TV5]  125:                Wandle(1, pos, false);
[UH6]  126:        #106 : if not Nurgross then
[VU9]  127:                Wandle(pos, length(str255),
           false);
[9B6]  128:         #75 : if pos > 1 then begin
[B84]  129:                pos := pred( pos );
[VMB]  130:                gotoxy( x1 + pred( pos ), y1
           );    { links }
[TUO]  131:               end;
[RSC]  132:         #77 : if pos < succ(length(str255))
           then begin
[NR9]  133:                gotoxy( x1 + pos, y1 );    {
           rechts }
```

```
[784]  134:                  pos := succ( pos );
[TUO]  135:                 end;
[7VB]  136:           #82 : if ins then begin
                       { Einfg  }
[VY2]  137:                  ins := false;
[G93]  138:                  Cursor(8,13);
[DLO]  139:                 end
[YS2]  140:                 else begin
[5G2]  141:                  ins := true;
[G93]  142:                  Cursor(1,13);
[TUO]  143:                 end;
[8PL]  144:           #83 : if (length(str255) > 0) and (
       length(str255) < succ(anzahl)) then
[BK7]  145:                  loeschChr( str255, pos, 1 );
[TUO]  146:         end;
[DLO]  147:        end
[YS2]  148:        else begin
[RS2]  149:         case ch of
[Z52]  150:          #27 : begin
[RF3]  151:                  str255[0] := #255;
[H23]  152:                  raus := true
[TUO]  153:                 end;
[EA6]  154:           #8 : if pos > 1 then begin
[B84]  155:                  pos := pred( pos );
[BK7]  156:                  loeschChr( str255, pos, 1 );
[TUO]  157:                 end;
[CU9]  158:          #13 : raus := true
                     { Return }
[A11]  159:         else
[1E6]  160:          if ch in bereich then
[MV4]  161:           if ins then begin
[CEB]  162:            if length(str255) < anzahl then be
       gin
[TN7]  163:             insert( ch, str255, pos );
[AU6]  164:             moveStr( x1, y1, str255 );
[056]  165:             gotoxy( x1 + pos, y1 );
[784]  166:             pos := succ( pos );
[TUO]  167:            end;
[DLO]  168:           end
[YS2]  169:           else begin
[HAA]  170:            if pos > length( str255 ) then beg
       in
[CEB]  171:             if length(str255) < anzahl then b
       egin
[B75]  172:              str255 := str255 + ch;
[670]  173:              p;
[TUO]  174:             end;
[DLO]  175:            end
[YS2]  176:            else begin
[NM4]  177:             str255[ pos ] := ch;
[670]  178:             p;
[TUO]  179:            end;
[TUO]  180:           end;
[TUO]  181:         end;
[TUO]  182:        end;
[R23]  183:      until raus;
[3Z3]  184:      Cursor(ca,ce);
[D48]  185:      RestoreWindow(5,17,70,9,hilf);
[TUO]  186:     end;
```

Listing 6. Diese Routine dient zum komfortablen Einlesen von Zeichenketten

Datum- und Zeiteingabe mit »Readdat.inc«

Die Include-Datei »Readdat.inc« (Listing 7) ist eine komfortable Leseroutine für Datum und Uhrzeit. Folgende Parameter werden benötigt:

x,y – Koordinaten des Einleseprozesses

datum – Datum im Record (siehe Listing)

zeit – Zeit im Record (siehe Listing)

time – »true«: Zeit wird eingelesen, sonst Datum

system – »true«: das Systemdatum und die Systemzeit werden zur Initialisierung verwendet

quit – »true«: Unterbrechung durch [Ctrl-C] oder [Esc] zugelassen.

Beim Lesen des Datums werden natürlich auch Schaltjahre berücksichtigt. Es ist nur die Eingabe der letzten beiden Ziffern der Jahreszahl notwendig, so daß die Jahreszahl zwischen 0 und 99 liegen darf. Die Zeiteingabe geschieht im Format von 0 bis 23 Uhr. Wurde die Prozedur durch [Ctrl-C] oder [Esc] abgebrochen, dann wird »quit« mit »true« belegt. Die Testroutinen, die die Richtigkeit des Datums und der Zeit prüfen, sind sehr aufwendig, aber dafür exakt. Das Editieren von Datum und Zeit geschieht der Einfachheit halber durch Überschreiben.

```
[R54]    0: {READDATEN.INC}
[26C]    1: ((Geändert für DOS international 7.1.88))
[000]    2:
[T85]    3: type TDatum = record
[TR4]    4:                  tag, monat : byte;
[BB4]    5:                       jahr : integer;
[TUO]    6:                 end;
[X94]    7:       TZeit = record
[LW2]    8:                  sekunde,
[LH2]    9:                   minute,
[Y84]   10:                    stunde : byte;
[TUO]   11:                 end;
[000]   12:
[4CN]   13:  procedure readDaten( x, y : byte; var datu
         m : tdatum; var zeit : tzeit;
[MNC]   14:                          time, system : boolea
         n; var quit : boolean );
[ABB]   15:   const s1 = 'Datumseingabe (tt.mm.jj) :';
[GC8]   16:         s3 = 'Zeiteingabe   (hh:mm:ss) :';
[82B]   17:         dh : array[ 1..12 ] of byte = ( 31,
         28, 31, 30, 31, 30, 31,
[S43]   18:                                          31,
         30, 31, 30, 31 );
[8Z7]   19:   type            tstr8 = string[ 8 ];
[5H8]   20:   var  j, posx, c, d, e : integer;
[GP3]   21:                   ca, ce : byte;
[D92]   22:                       ch : char;
[LN3]   23:                   Cstr8 : tstr8;
[L24]   24:                    str1 : string[1];
[5T2]   25:                    str8 : tstr8;
[G94]   26:                    str2 : string[ 2 ];
[9Z6]   27:           esc, raus, ok : boolean;
[000]   28:
[9R4]   29:   procedure err;
[BH1]   30:     begin
[WH2]   31:      signal;
[9R5]   32:      MoveStr(x,y,Cstr8);
[N62]   33:      posx := 1;
[A74]   34:      gotoxy( x, y );
[TUO]   35:     end;
[000]   36:
[FXD]   37:   procedure datumlesen( var datum : tdatum
         );
[Z65]   38:    var regs : tregs;
[BH1]   39:     begin
[FF3]   40:      with regs do
[H5A]   41:       with datum do begin   (Systemdatum)
[XG1]   42:        ah := $2A;
[W93]   43:        msdos(regs);
[YF3]   44:        jahr := cx-1900;
[ZX2]   45:        monat := dh;
[XT1]   46:        tag := dl;
[EZ4]   47:        (wochentag := al;)
[TUO]   48:       end;
[TUO]   49:     end;
[000]   50:
[29C]   51:   procedure zeitlesen( var zeit : tzeit );
[Z65]   52:    var regs : tregs;
[BH1]   53:     begin
[FF3]   54:      with regs do
[BVA]   55:       with zeit do begin (Systemzeit)
[ZG1]   56:        ah := $2C;
[W93]   57:        msdos(regs);
[HG3]   58:        stunde := ch;
[WK3]   59:        minute := cl;
[1D4]   60:        sekunde := dh;
[FF3]   61:        (frac := dl;)
[TUO]   62:       end;
[TUO]   63:     end;
[000]   64:
[BH1]   65:    begin
[ER3]   66:     if time then
[HS4]   67:      MoveStr(x,y,s3)
[A11]   68:     else
[5S4]   69:      MoveStr(x,y,s1);
[N62]   70:     posx := 1;
[TW3]   71:     raus := false;
[L92]   72:     esc := QUIT;
[OX3]   73:     quit := false;
[AC2]   74:     Cstr8 := '';
[2F5]   75:     if time then begin
[D55]   76:      if not system then
[GG3]   77:       Cstr8 := '00:00:00'
[YS2]   78:      else begin
[KC5]   79:       zeitlesen(zeit);
[2HE]   80:       str(zeit.stunde:2,str2);  Cstr8 := co
         py(str2,1,2);
[HMH]   81:       str(zeit.minute:2,str2);  Cstr8 := Cs
         tr8 + ':' + copy(str2,1,2);
[TVH]   82:       str(zeit.sekunde:2,str2); Cstr8 := Cs
         tr8 + ':' + copy(str2,1,2);
[TUO]   83:      end;
```

```
[DLO]   84:      end
[A11]   85:      else
[D55]   86:      if not system then
[ME3]   87:       Cstr8 := '00.00.00'
[YS2]   88:      else begin
[UV5]   89:       datumlesen(datum);
[U8D]   90:       str(datum.tag:2,str2);    Cstr8 := cop
          y(str2,1,2);
[1EH]   91:       str(datum.monat:2,str2);  Cstr8 := Cst
          r8 + '.' + copy(str2,1,2);
[XXG]   92:       str(datum.jahr:2,str2);   Cstr8 := Cst
          r8 + '.' + copy(str2,1,2);
[TUO]   93:      end;
[HUB]   94:      while pos(#32,Cstr8) > 0 do      {Durch
          SYSTEM}
[UH6]   95:       Cstr8[pos(#32,Cstr8)] := #48;
[ZF3]   96:      str8 := Cstr8;
[ML5]   97:      x := x + length(s1)+2;
[7N5]   98:      CursorFrage(0,ca,ce);
[9R5]   99:      MoveStr(x,y,Cstr8);
[893]  100:      gotoxy(x,y);
[X52]  101:      repeat
[862]  102:       read2( ch );
[UY5]  103:       if ch = #0 then begin
[862]  104:        read2( ch );
[RS2]  105:        case ch of
[M27]  106:         #75 : if posx > 1 then begin
[WS6]  107:                if posx in [ 4, 7 ] then
[3U3]  108:                 posx := posx - 2
[A11]  109:                else
[735]  110:                 posx := pred( posx );
[A6C]  111:                gotoxy( x + pred( posx ), y )
          ;       { links  }
[TUO]  112:               end;
[518]  113:         #77 : if  posx < 8 then begin
[ES6]  114:                if posx in [ 2, 5 ] then
[RT3]  115:                 posx := posx + 2
[A11]  116:                else
[635]  117:                 posx := succ( posx );
[KHC]  118:                gotoxy( x + pred( posx ), y )
          ;       { rechts }
[TUO]  119:               end;
[TUO]  120:       end;
[DLO]  121:      end
[YS2]  122:      else begin
[RS2]  123:       case ch of
[CL6]  124:         #3 : if esc then begin
[SB3]  125:                raus := true;
[ZB3]  126:                quit := true;
[WH2]  127:                signal;
[TUO]  128:               end;
[E08]  129:         #32 : if  posx < 8 then begin
[ES6]  130:                if posx in [ 2, 5 ] then
[RT3]  131:                 posx := posx + 2
[A11]  132:                else
[635]  133:                 posx := succ( posx );
[KHC]  134:                gotoxy( x + pred( posx ), y )
          ;       { rechts }
[TUO]  135:               end;
[FU6]  136:         #27 : if esc then begin
[SB3]  137:                raus := true;
[ZB3]  138:                quit := true;
[WH2]  139:                signal;
[TUO]  140:               end;
[H52]  141:         #13 : begin
[8H5]  142:                str2 := copy( str8, 1, 2 );
[W84]  143:                val( str2, j, c );
[ER3]  144:                if time then
[8B4]  145:                 zeit.stunde := j
[A11]  146:                else
[TD4]  147:                 datum.tag := j;
[RH5]  148:                str2 := copy( str8, 4, 5 );
[394]  149:                val( str2, j, d );
[ER3]  150:                if time then
[Y84]  151:                 zeit.minute := j
[A11]  152:                else
[X94]  153:                 datum.monat := j;
[6K5]  154:                str2 := copy( str8, 7, 8 );
[A94]  155:                val( str2, j, e );
[ER3]  156:                if time then
[CP4]  157:                 zeit.sekunde := j
[A11]  158:                else
[WE4]  159:                 datum.jahr := j;
[ER3]  160:                if time then
[XLB]  161:                 if not( zeit.stunde in [0..2
          3] ) or ( c <> 0 ) or
[7VB]  162:                    not( zeit.minute in [0..5
          9] ) or ( d <> 0 ) or
[VOC]  163:                    not( zeit.sekunde in [0..
          59] ) or ( e <> 0 ) then
[ZMO]  164:                  err
[A11]  165:                 else
[H23]  166:                  raus := true
[A11]  167:                else
[WMB]  168:                if not( datum.tag in [1..31]
          ) or ( c <> 0 ) or
[3FB]  169:                   not( datum.monat in [1..12
          ] ) or ( d <> 0 ) or
[Y1C]  170:                   not( datum.jahr in [80..99
          ] ) or ( e <> 0 ) then
[ZMO]  171:                 err
[YS2]  172:                else begin
[01B]  173:                 if datum.tag > dh[ datum.mon
          at ] then
[N16]  174:                  if ( datum.monat = 2 ) and
[22D]  175:                     ( ( datum.jahr - 76 ) mo
          d 4 = 0 ) then (Schaltjahr)
[UD6]  176:                   if datum.tag = 29 then
[H23]  177:                    raus := true
[A11]  178:                   else
[ZMO]  179:                    err
[A11]  180:                  else
[ZMO]  181:                   err
[A11]  182:                 else
[SB3]  183:                  raus := true;
[GU3]  184:                 if raus then
[6B8]  185:                  datum.jahr := datum.jahr +
          1900
[A11]  186:                 else
[WH2]  187:                  signal;
[TUO]  188:                end;
[DLO]  189:       end
[A11]  190:       else
[HY7]  191:       if ch in ['0'..'9'] then begin
[NN2]  192:        ok := false;
[ER3]  193:        if time then
[SF3]  194:         case posx of
[Y76]  195:                1 : ok := ( ch in ['0'..'2']
          );
[H97]  196:             2, 5, 8 : ok := ( ch in ['0'..'9']
          );
[NW5]  197:                4, 7 : ok := ( ch in ['0'..'5']
          );
[DLO]  198:         end
[A11]  199:         else
[SF3]  200:         case posx of
[086]  201:                1 : ok := ( ch in ['0'..'3']
          );
[H97]  202:             2, 5, 8 : ok := ( ch in ['0'..'9']
          );
[HW5]  203:                4 : ok := ( ch in ['0','1']
          );
[U96]  204:                7 : ok := ( ch in ['8'..'9']
          );
[TUO]  205:         end;
[ZX4]  206:         if ok then begin
[AF4]  207:          str8[ posx ] := ch;
[ZF7]  208:          MoveChr(x+pred(posx),y,ch);
[YU8]  209:          if posx in [ 2, 5 ] then begin
[C15]  210:           posx := succ(posx);
[H66]  211:           gotoxy( x + posx, y );
[C15]  212:           posx := succ(posx);
[DLO]  213:          end
[A11]  214:          else
[577]  215:          if posx < 8 then begin
[H66]  216:           gotoxy( x + posx, y );
[635]  217:           posx := succ( posx );
[TUO]  218:          end;
[TUO]  219:         end;
[TUO]  220:        end;
[TUO]  221:       end;
[TUO]  222:      end;
[R23]  223:     until raus;
[3Z3]  224:     Cursor(ca,ce);
[TUO]  225:    end;
```

Listing 7. Quellcode der Include-Datei zur Datums- und Zeiterfassung

Komfortable Eingabe von Integer-Zahlen

Die Prozedur »ReadInt« (Listing 8) liest eine Integer-Zahl ein. Die Routine arbeitet mit den Parametern:

wert – der einzulesende Integer-Wert

min – die kleinste zulässige Zahl beim Einlesen

max – die größte zulässige Zahl beim Einlesen

escR – »true«: Die Taste [Esc] führt zum Abbruch der Eingabe

Diese Prozedur ruft die Include-Datei »ReadStr.inc« auf. Wurde die Prozedur mit [Esc] abgebrochen, so wird »escR« mit »True« belegt.

(Listing 8 siehe fogende Seite.)

```
[5A3]    0: (READINT.INC)
[06C]    1: ((Erstellt für DOS international 7.1.88))
[000]    2:
[OFN]    3: procedure readint( var wert : integer; min,
            max : integer; x, y : byte;
[B06]    4:                     var escR : boolean );
[44A]    5:  (Wert muß initialisiert sein)
[AZB]    6:  const zugelassen : tsetofchar = ['0'..'9']
           ;
[K97]    7:  var   maxStellen : byte;
[K15]    8:            Merk, c : integer;
[185]    9:            esc, ok : boolean;
[A44]   10:               str6 : string[6];
[BH1]   11:   begin
[MA4]   12:    str( max, str6 );
[ZB2]   13:    esc := escR;
[E93]   14:    Merk := Wert;
[NP7]   15:    maxStellen := length( str6 );
[D86]   16:    if min < 0 then begin
[2P9]   17:     maxStellen := succ( maxStellen );
[2Z9]   18:     zugelassen := zugelassen + ['-','+'];
[TUO]   19:    end;
[X52]   20:    repeat
[RK4]   21:     str(Wert,str6);
[XSE]   22:     readstr(str6,zugelassen,maxstellen,x,y,
          true);
[NN2]   23:     ok := false;
[A75]   24:     if str6[0] <> #0 then
[KU7]   25:      if str6[0] <> #255 then begin
[6U5]   26:       val( str6, wert, c );
[19F]   27:       if ( c <> 0 ) or ( wert > max ) or (
          wert < min ) then begin
[WH2]   28:        signal;
[E93]   29:        Wert := Merk;
[DLO]   30:       end
[A11]   31:       else
[UC2]   32:        ok := true;
[DLO]   33:      end
[A11]   34:      else
[XS4]   35:       if esc then begin
[UC2]   36:        ok := true;
[N63]   37:        escR := true;
[TUO]   38:       end;
[HL2]   39:    until ok;
[TUO]   40:   end;
```

Listing 8. Turbo Pascal-Routine zur Eingabe von Integer-Werten

```
[EXM]    0: { Packed - Emulation unter Turbo 22.6.87 (f
            ür DOS international 7.1.88)
[WHL]    1:   Felder bis zu 32767 Byte können bitweise
            (!!!) gepacked/ausgepackt
[PF2]    2:   werden.
[GUR]    3:   Der erste Parameter ist die Quelle, die g
            epackt/ausgepackt werden soll,
[SML]    4:   und der Zweite das Ziel; "anz" gibt die A
            nzahl der zu packenden/
[T28]    5:   auszupackenden Bytes an }
[000]    6:
[57F]    7:  procedure pack( var st1; var st2; anz : in
            teger );
[PT4]    8:   var i, h : integer;
[BH1]    9:    begin
[DB7]   10:     for i := 0 to pred( anz ) do
[769]   11:      memw[ seg( st2 ) : ofs( st2 ) + i ] :=
            0;
[YY8]   12:     for i := 0 to pred(anz) do begin
[DX2]   13:      h := ( i shr 3 );
[W2C]   14:      if mem[ seg( st1 ) : ofs( st1 ) + i ]
            = 1 then
[8W7]   15:       mem[ seg( st2 ) : ofs( st2 ) + h ] :=
[15B]   16:       mem[ seg( st2 ) : ofs( st2 ) + h ] +
            1 shl ( i mod 8 );
[TUO]   17:     end;
[TUO]   18:    end;
[000]   19:
[2VF]   20:  procedure auspack( var st1; var st2; anz :
            integer );
[Y56]   21:   var h, j, k, i : integer;
[BH1]   22:    begin
[YY8]   23:     for i := 0 to pred(anz) do begin
[L64]   24:      k := 1 shl ( i mod 8 );
[FX2]   25:      j := ( i shr 3 );
[YAB]   26:      h := ( mem[ seg( st2 ) : ofs( st2 ) +
            j ] and k ) ;
[X03]   27:      if h = k then
[SC8]   28:       mem[ seg( st1 ) : ofs( st1 ) + i ] :=
            1
[A11]   29:      else
[HC8]   30:       mem[ seg( st1 ) : ofs( st1 ) + i ] :=
            0;
[TUO]   31:     end;
[TUO]   32:    end;
[000]   33:
```

Listing 9. Quellcode der Pack-Routine »pack.inc«

Bitweise gepackt mit Turbo Pascal

Im Turbo Pascal-Handbuch ist nachzulesen, daß Turbo Pascal automatisch packt. Leider verschweigt es dem Leser, daß die Programmierer nicht das übliche Verfahren des bitweisen, sondern des byteweisen Packens verwenden. Programmierer, die mit UCSD-Pascal arbeiten, haben es da schon besser. Die in den meisten anderen Pascal-Dialekten vorkommenden Routinen »Pack« und »Unpack« lassen sich leicht selbst programmieren. Folgende Routinen (»pack.inc«) packen boolsche Felder in Byte-Felder, so daß sich der Speicherbedarf um das Achtfache reduziert (Listing 9):

Pack(st1, st2, anz); »st1« und »st2« sind typenlose Parameter, weshalb die Felder beliebig groß sein dürfen. Der Parameter »st1« ist das zu packende Feld, während »st2« das Zielfeld angibt. Die Anzahl der zu packenden Boolean (im Zweifelsfall mit »sizeof« abfragen) wird mit dem Parameter »anz« übergeben.

Auspack(st1, st2, anz); Diese Prozedur arbeitet wie »pack«, nur mit dem Unterschied, daß »st1« das Zielfeld darstellt. Um ein 32 KByte langes Feld von booleschen Variablen zu packen, benötigt diese Routine zirka fünf Sekunden. Das Entpacken dauert etwa genauso lange. Das Demo-Programm (Listing 10) demonstriert die Fähigkeiten der Include-Datei »pack.inc«.

```
[38F]    0: ((Erstellt für DOS international 7.1.88) (c
            ) A.Baqué)
[E45]    1: {$I-,R-,C-,U-,D-,V-,K-}
[ZOA]    2:  var  a1, a4 : array[1..16384] of boolean;
[RD6]    3:           a2 : array[1..2048] of byte;
[V94]    4:       a3, a5 : integer;
[UD3]    5:           ok : boolean;
[000]    6:
[G13]    7:  {$I PACK.INC}
[000]    8:
[BH1]    9:   begin
[5LB]   10:    for a3 := 1 to 16384 do (Initialisierung
            )
[KP4]   11:     if odd(a3) then
[DX3]   12:      a1[a3] := true
[A11]   13:     else
[9H4]   14:      a1[a3] := false;
[HN7]   15:    move(a1[1],a4[1],sizeof(a1));
[RC6]   16:    writeln(#7,'Packen...');
[X95]   17:    pack(a1[1],a2[1],16384);
[9H8]   18:    writeln(#7,'Taste drücken');
[P81]   19:    read;
[LMK]   20:    fillchar(a1[1],16384,chr(0));  {Zerstöre
          alte Ordnung für Text}
[4L7]   21:    writeln(#7,'Auspacken...');
[G39]   22:    auspack(a1[1],a2[1],sizeof(a1));
[372]   23:    write(#7);
[RA1]   24:    a3 := 1;
[UC2]   25:    ok := true;
[NYG]   26:    while (a3 <= 16384 ) and ok do begin  {T
          EST, ob es geklappt hat}
[MH5]   27:     ok := ( a1[a3] = a4[ a3 ] );
[WE3]   28:     a3 := succ(a3);
[TUO]   29:    end;
[GRD]   30:    writeln(ok);  {TRUE: Alles in Ordnung, F
          ALSE : SCHIT}
[5TO]   31:   end.
```

Listing 10. »packdemo.pas« demonstriert die Fähigkeiten der Pack-Routine

Mehr Übersicht durch Pulldown-Menüs

Pulldown-Menüs machen Programme leichter verständlich und vereinfachen die Bedienung. Im folgenden werden zwei verschiedene Arten gezeigt. Für ein eindimensionales Pulldown-Menü müssen Sie lediglich die Menüpunkte auf den Bildschirm bringen und die Prozedur »Wahl« (»pulldow0.inc«, Listing 11) aufrufen. Diese hat folgende Syntax:

Wahl(x, y, anzahl1, anzahl2, break); »x,y« sind die Koordinaten des ersten Menüpunkts, wobei dem Parameter »y« der Wert des durch [Return] angewählten Menüpunkts übergeben wird. »anzahl1« gibt die maximale Länge der Menüpunkte an, während »anzahl2« die Anzahl der Menüpunkte enthält. Ist der Parameter »break« gleich »true«, so kann die Routine durch die Taste [Esc] unterbrochen werden. Außerdem verwendet die Prozedur »Wahl« die Routine »Invers2«, die auch in anderen Programmen Verwendung finden könnte:

Invers2(x, y, anzahl, inv); Diese Prozedur invertiert beliebige Bildschirmbereiche (Attribut 15/112) und benötigt folgende Parameter:

x,y – Koordinaten des zu invertierenden Bereichs
anzahl – Anzahl der zu invertierenden Positionen
inv – Ist »inv« gleich »true«, dann wird das Attribut »112« verwendet, sonst »15« (siehe »Library«).

Komfortabler wird die Bedienung mit zweidimensionalen Pulldown-Menüs (Listing 12). Programme wie »Open Acces« verwenden diese zweidimensionalen Pulldown-Menüs ebenfalls. Wer bereits mit diesem Programm gearbeitet hat, wird diese Art der Menüführung zu schätzen wissen. Wie bei der Funktion »Wahl« müssen die Menüpunkte erst auf den Bildschirm geschrieben werden. Anschließend erfolgt der Aufruf der Wahl2-Routine:

```
Wahl2( a, x, y, x1, y1, test2, minus, anz, xanz, yanz, raus );
```

Die Prozedur »Wahl2« arbeitet mit folgendem Parameter:

a – rettet letztes Attribut
x,y – Koordinate des Menüpunktes in der linken oberen Ecke
x1,y1 – Koordinate beim Verlassen von »Wahl2«
test2 – zweite Position, die »Wahl2« testet, ob Menüpunkt existiert
minus – erste Position, die »Wahl2« testet, ob Menüpunkt existiert
anz – Maximale Länge der Menüpunkte
xanz – Anzahl der Menüpunkte in x-Richtung
yanz – Anzahl der Menüpunkte in y-Richtung
raus – Gibt an, ob Prozedur mit [Esc] beendet wurde

Interessant ist hier, daß »Wahl2« testet, ob der Menüpunkt überhaupt existiert (»test2«, »minus«). Wie bei der Prozedur »Wahl« gibt es auch hier eine Hilfsroutine:

```
Invers1( att, x, y, anzahl, inv );
```

Beim Invertieren mit dem Attribut »112« wird der Variablen »att« das alte Attribut übergeben. Außerdem invertiert die Routine nur Stellen, die nicht mit Leerzeichen (chr(32)) gefüllt sind. Ansonsten ist die Prozedur identisch mit »Invers2«.

```
[6WG]    0: {Pulldown-Menu (Geandert fur DOS internatio
           nal 8.1.88) }
[000]    1:
[1ME]    2: { ---------- Eindimensionales Pulldownmenü
           --------------- }
[A2H]    3:  procedure invers2( x, y, anzahl : byte; in
           v : boolean);
[W63]    4:   var i : byte;
[BH1]    5:    begin
[ZV2]    6:     if inv then
[X94]    7:      attribut := 112
[A11]    8:     else
[MC4]    9:      attribut := 15;
[9F2]   10:     Adr(x,y);
[1L8]   11:     for i := 0 to pred( anzahl ) do
[R2G]   12:       mem[ screenaddr : succ(memadr+ (i) shl
           1) ] := attribut;
[TU0]   13:    end;
[000]   14:
[DFM]   15:  procedure wahl( x : byte; var y : byte; an
          zahl1, anzahl2 : byte;
[345]   16:                  break : boolean );
[W63]   17:   var i : byte;
[D92]   18:       ch : char;
[BH1]   19:    begin
[MD1]   20:     i := 2;
[R78]   21:     invers2( x, y, anzahl1, true );
[X52]   22:     repeat
[862]   23:      read2( ch );
[YM5]   24:      if ( ch = #0 ) then begin
[862]   25:       read2( ch );
[ST4]   26:       case ord(ch) of
[929]   27:        80 : if i <= anzahl2 then begin
[9C8]   28:               invers2( x, y, anzahl1, false
          );
[AS3]   29:               y := succ( y );
[UN3]   30:               i := succ( i );
[R78]   31:               invers2( x, y, anzahl1, true )
          ;
[TU0]   32:             end;
[R76]   33:        72 : if i > 2 then begin
[9C8]   34:               invers2( x, y, anzahl1, false
          );
[4R3]   35:               y := pred( y );
[MM3]   36:               i := pred( i );
[R78]   37:               invers2( x, y, anzahl1, true )
          ;
[DL0]   38:             end
[A11]   39:             else
[7K5]   40:              if break then begin
[ED1]   41:               i := 1;
[4R3]   42:               y := pred( y );
[TU0]   43:             end;
[TU0]   44:       end;
[TU0]   45:      end;
[ST5]   46:      if break and ( i = 1 ) then
[AG1]   47:       ch := #13;
[CT3]   48:     until ch = #13;
[MC4]   49:     attribut := 15;
[TU0]   50:    end;
[000]   51:
[FXG]   52: { ---------- Ende Eindimensionales Pulldown
          menü --------------- }
```

Listing 11. Quellcode der Include-Datei für eindimensionale Pull-Down-Menüs (»pulldow0.inc«)

```
[4WP]    0: {2-dimensionales Pulldown-Menü (Geändert fü
           r DOS international 8.1.88)}
[000]    1:
[2AH]    2:  procedure invers1( var att : byte; x, y, a
           nzahl : byte;
[A34]    3:                     inv : boolean);
[W63]    4:   var i : byte;
[BH1]    5:    begin
[9F2]    6:     Adr(x,y);
[4K6]    7:     MemAdr := succ(MemAdr);
[6W4]    8:     if inv then begin
[X94]    9:      attribut := 112;
[2S8]   10:      att := Mem[screenaddr:Memadr];
[DL0]   11:     end
[A11]   12:     else
[DC5]   13:      attribut := att;
[H08]   14:     for i := 1 to anzahl do begin
[ZCD]   15:      if mem[ screenaddr : pred(MemAdr) ] <>
          32 then
[31C]   16:       mem[ screenaddr : MemAdr ] := attribu
          t;
[XV4]   17:      MemAdr := MemAdr+2;
[TU0]   18:     end;
[TU0]   19:    end;
[000]   20:
[9HB]   21:  procedure wahl2( var a, x, y, x1, y1 : byt
          e;
[ZLG]   22:                   test2, minus, anz, xanz,
          yanz : byte; var raus : boolean );
[F84]   23:   var   ch : char;
[UD3]   24:         ok : boolean;
[000]   25:
[OKA]   26:   function test( x, y : byte ) : boolean;
[BH1]   27:     begin
[DEG]   28:      test := ((mem[screenaddr:(y*5) shl 5+p
          red(x) shl 1 ] <> 32) or
```

```
[S4D]   29:                (mem[screenaddr:(y*5) shl 5+(
          x+test2) shl 1] <> 32))
[TUO]   30:      end;
[000]   31:
[KG6]   32:    procedure g( h : byte );
[DS2]   33:      begin  { g }
[83H]   34:       invers1( a, x, y, anz, false );      {
          unten : ohne pred, da 1 nicht 01 }
[832]   35:       case h of
[H67]   36:        1 : if test(x,y) then begin
[963]   37:              y := succ(y);
[SF3]   38:              y1 := succ( y1 );
[TUO]   39:            end;
[E98]   40:        2 : if test(x,y-2) then begin
[353]   41:              y := pred(y);
[7F3]   42:              y1 := pred( y1 );
[TUO]   43:            end;
[6RB]   44:        3 : if test(x-minus,pred(y)) then beg
          in
[F94]   45:              x := x - minus;
[1F3]   46:              x1 := pred( x1 );
[TUO]   47:            end;
[VPB]   48:        4 : if test(x+minus,pred(y)) then beg
          in
[F94]   49:              x := x + minus;
[KF3]   50:              x1 := succ( x1 );
[TUO]   51:            end;
[B92]   52:        6 : begin
[X52]   53:              repeat
[F94]   54:               x := x - minus;
[GH3]   55:               x1 := pred(x1);
[UA3]   56:              until x1 = 1;
[963]   57:              y := succ(y);
[6K3]   58:              y1 := succ(y1);
[TUO]   59:            end;
[A92]   60:        5 : begin
[X52]   61:              repeat
[F94]   62:              x := x + minus;
[1K3]   63:              x1 := succ(x1);
[Y85]   64:              until x1 = xanz;
[353]   65:              y := pred(y);
[NH3]   66:              y1 := pred(y1);
[TUO]   67:            end;
[TUO]   68:       end;
[FB7]   69:       invers1( a, x, y, anz, true );
[T13]   70:      end;      { g }
[000]   71:
[L7S]   72: {Individuell können folgende Procedur, sowi
          e der Hauptteil von WAHL2 aussehen
[RL7]   73:  procedure go( ch : byte );
[BH1]   74:    begin
[RS2]   75:     case ch of
[B26]   76:       80 : if y1 < yanz then
[NSO]   77:              g(1);
[SS3]   78:       72 : if y1 > 1 then
[SSO]   79:              g(2);
[XS3]   80:       75 : if x1 > 1 then
[MFO]   81:              g(3)
[A11]   82:            else
[UP3]   83:            if y1 > 1 then
[1TO]   84:              g(5);
[H26]   85:       77 : if x1 < xanz then
[RFO]   86:              g(4)
[A11]   87:            else
[XE4]   88:            if y1 < yanz then
[4TO]   89:              g(6);
[TUO]   90:     end;
[TUO]   91:    end;
[000]   92:
[BH1]   93:    begin
[TW3]   94:     raus := false;
[NN2]   95:     ok := false;
[FB7]   96:     invers1( a, x, y, anz, true );
[X52]   97:     repeat
[862]   98:      read2( ch );
[YM5]   99:      if ( ch = #0 ) then begin
[862]  100:       read2( ch );
[K97]  101:       if ch in [#80,#72,#75,#77] then
[4U2]  102:        go(ord(ch));
[DLO]  103:      end
[YS2]  104:      else begin
[YE4]  105:       ch := upcase(ch);
[RS2]  106:       case ch of
[K54]  107:        #13 : ok := true;
[W24]  108:        #27 : raus := true;
[TUO]  109:       end;
[TUO]  110:      end;
[S75]  111:     until ok or raus;
[AE1]  112:    end;)
```

Listing 12. Quellcode der Include-Datei für zweidimensionale Pull-Down-Menüs (»pulldow1.inc«)

Inhaltsverzeichnis mal ganz anders

Sie kennen sicher die Problematik: In einem eigenen Programm soll das Inhaltsverzeichnis eines Datenträgers eingelesen werden. Das DOS-Kommando »Dir« ist für diese Fälle meist nicht geeignet. Hier finden Sie vier Programme, die alle Forderungen in dieser Richtung abdecken. Die Routinen »Dir4«, »Dir6«, »Mdir« und »Sdir« lassen sich sowohl in eigenen Programmen einsetzen, als auch als eigenständige Utilities verwenden. Um eines dieser Programme als Prozedur zu nutzen, reicht folgende Einbindung:

```
procedure Dir;
{$I Dir6} {oder Dir4, Mdir, Sdir}«
```

Diese Zeilen machen das Hauptprogramm zum Programm der Prozedur »Dir«. Natürlich müssen die in den Programmen verwendeten Include-Dateien vom Hauptprogramm aufgerufen werden. Alle diese Programme sind schneller und komfortabler als das System-Directory. Besonders komfortabel sind die Programme »Dir6« und »Mdir«. Jedes hat seine spezifischen Eigenschaften, die sie jeweils für einen speziellen Fall eignen. Es werden jedoch immer alle Dateien eingelesen und eine Speicherstatistik wird ausgegeben. Außerdem zeigen die vier Programme den Datenträgernamen, die Dateianzahl, die Verzeichnisgröße und den Suchpfad an. Beim Abtippen können Sie sich jedoch einiges an Arbeit ersparen, da viele Programmabschnitte identisch oder ähnlich sind. Speziell das Programm »Sdir« bildet die Grundlage für das später vorgestellte Programm »Execute«.

Jede hier vorgestellte Directory-Routine eignet sich für ein anderes Einsatzgebiet. Das Programm »Dir4« (Listing 13) liefert ein Inhaltsverzeichnis, das dem DOS-Befehl »Dir« ähnlich sieht. Die einzelnen Felder werden jedoch im Gegensatz dazu durch Trennstriche geteilt (Bild 3). Dieses Programm benötigt die Include-Dateien »Library«, »LibraryE«, »Window« und »Tools«.

Die Directory-Routine »Dir6« (Listing 14) gibt das Inhaltsverzeichnis in einem Bildschirmfenster aus (Bild 4). Der Bildschirmhintergrund geht dadurch nicht verloren. Dieser Vorgang geschieht sehr schnell. Auf einen Druck der Return-Taste steht innerhalb 1/100 Sekunde die erste Seite der Directory-Einträge komplett auf dem Bildschirm. Dieses Programm benutzt die Include-Dateien »Library«, »LibraryE«, »Window«, »Tools« und »ReadStr«.

»Sdir« (Listing 15) ist ein Directory-Programm in Gestalt eines zweidimensionalen Pulldown-Menüs. Mit den Cursor-Steuertasten und der Return-Taste können Sie den Verzeichnispfad »durchwandern«. »Sdir« verfügt außerdem über eine Filter-, eine Statistik- und eine Sortierfunktion und kann auch ein neues Laufwerk bestimmen. Beim

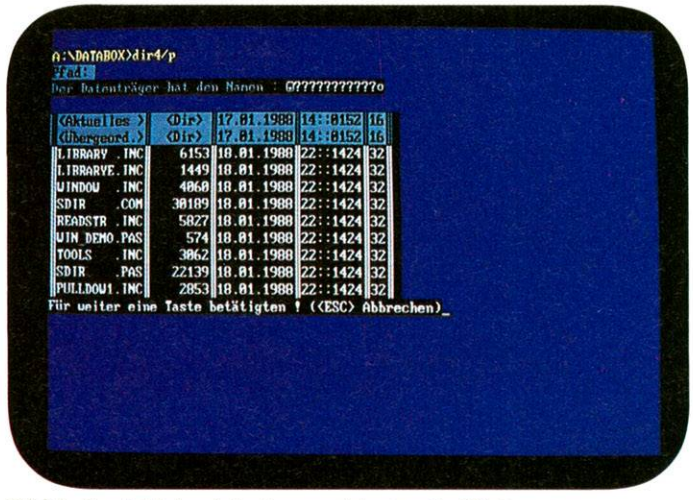


Bild 3. So sieht das Inhaltsverzeichnis mit »Dir4« aus

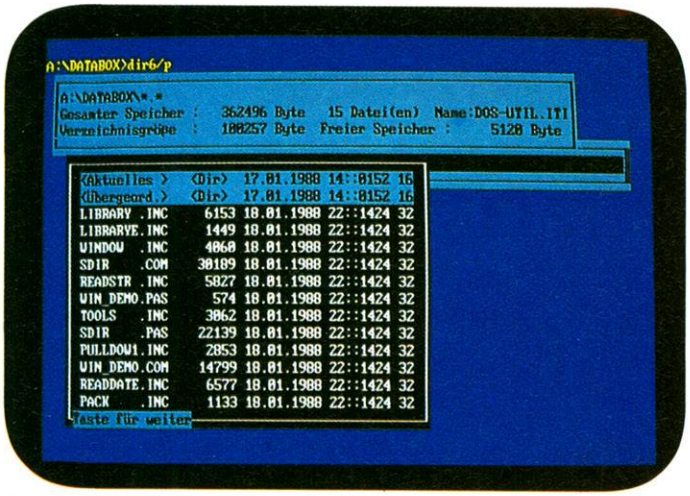


Bild 4. Directory-Anzeige mit »Dir6«

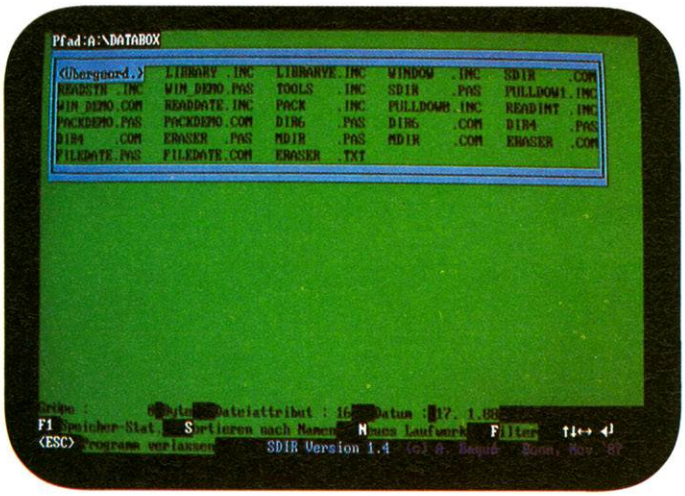


Bild 5. Directory mit Window-Ausgabe

Start liest »Sdir« immer das Root-Directory, es ist aber sehr einfach, sich durch den Verzeichnisbaum zu bewegen (Bild 5). Als Include-Dateien benötigt es dieselben wie »Dir6«. Für die Darstellung des Pulldown-Menüs benötigt »Sdir« zusätzlich die Datei »Pulldow1.inc«. Optisch am besten gelungen ist das Programm »Mdir« (Listing 16).

Die Directory-Routine ist sehr schnell und mit einer Sortier- und Blätterfunktion (vorwärts und rückwärts) ausgerüstet. Auch dieses Programm arbeitet mit Window-Technik (Bild 6). Im Vergleich zu »Dir6« ist es aber völlig anders aufgebaut. »Mdir« benötigt dieselben Include-Dateien wie »Dir6«.

Die Programme »Dir4«, »Dir6« und »Mdir« akzeptieren Parameter von »/1« bis »/7« (Bild 7). Die Standardeinstellung für alle Directory-Befehle ist »/7«. Der Parameter »/s« aktiviert bei »Mdir« die Sortierfunktion. Um mit »Dir4« seitenweise zu listen, müssen Sie wie beim System-Befehl »Dir« den Parameter »/p« angeben.

Die anderen Programme tun dies auch ohne Aufforderung. Lassen Sie bei den Programmen »Dir6« und »Mdir« die Parameter-Angabe fort, so ist die kontrollierte Eingabe im Programm erlaubt.

Alle Funktionen, die unter DOS angewendet werden können, sind übrigens auch bei diesen Programmen anwendbar. Eine komplizierte Auswertungs-Routine wertet Eingaben und Systeminformationen perfekt aus.

Parameter	Name	Größe	Datum	Zeit	Attribut
/1	ja	nein	nein	nein	nein
/2	ja	ja	nein	nein	nein
/3	ja	nein	ja	nein	nein
/4	ja	nein	ja	ja	nein
/5	ja	ja	ja	nein	nein
/6	ja	ja	ja	ja	nein
/7	ja	ja	ja	ja	ja
/8 nur Mdir	ja	ja	ja	ja	ja, Text !

Bild 7. Die Parameter der Directory-Befehle

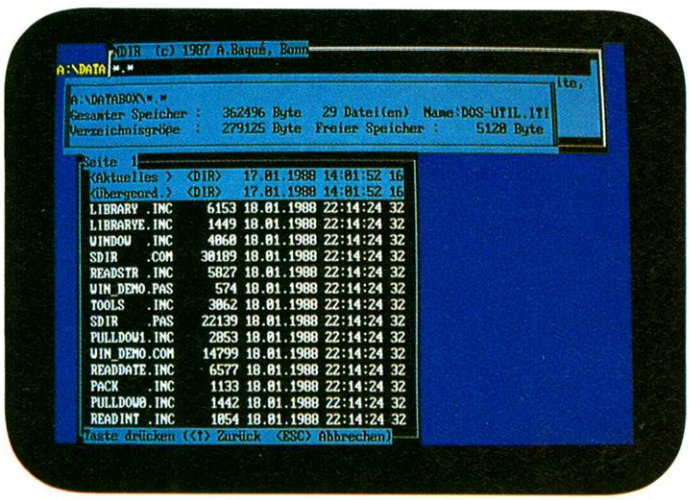


Bild 6. »Mdir« zeigt das Inhaltsverzeichnis an

```
[375]    0: {$I-,K-,D-,R-,V-,C-,U-}
[000]    1:
[NAF]    2: {DIR4.PAS 28.8.87, (Geändert für DOS intern
           ational 8.1.88)}
[000]    3:
[WP3]    4: {$I LIBRARY.INC}
[R64]    5: {$I WINDOW.INC}
[XU3]    6: {$I LIBRARYE.INC}
[4P3]    7: {$I TOOLS.INC}
[000]    8:
[2P4]    9: procedure dir;
[BE9]   10:  const          optionen : integer = 7;
[V4C]   11:  type                          ast = array[1
           ..43] of byte;
[LD5]   12:                              tstr12 = string[
           12];
[FS6]   13:  var                            ch : Char;
[5R3]   14:                      anzahl, y, q1,
[D17]   15:                    q2, anzxss, i, c : integer
           ;
[W99]   16:      speicherBelegt, speicherFrei,
[Z88]   17:                groesse, speicherKap : real;
[N62]   18:                                 as : ast;
[B54]   19:                               str9 : string[
           9];
[Y24]   20:                               str2 : string[
           2];
[5Z3]   21:                             str255 : tstr255
           ;
[HG5]   22:                   t, c5, a, b, d, e : byte;
[OX5]   23:            paging, gef, b1, b2, b3,
[XV7]   24:                  exist, error, raus : boolean
           ;
[AT3]   25:                               str80 : tstr80;
[3W2]   26:                                  zw : tstr12;
[T14]   27:                                 zw1 : string[
           13];
[X33]   28:                               regs : tregs;
[000]   29:
[6A5]   30:  procedure hoch;
[BH1]   31:    begin
[RW5]   32:      if y >= 25 then begin
[SCC]   33:        rollen1; {Schneller als BIOS-Routine !
         }
```

```
[UC1]   34:       y := 25;
[DL0]   35:     end
[A11]   36:     else
[963]   37:       y := succ(y);
[652]   38:     adr(1,y);
[TU0]   39:    end;
[000]   40:
[TOC]   41:  procedure weiter( var raus : boolean );
[5L3]   42:   var    ch : char;
[BH1]   43:    begin
[MC4]   44:     attribut := 15;
[75M]   45:     SchreibeSatz('Für weiter eine Taste bet
        ätigten ! (<ESC> Abbrechen)');
[PP2]   46:     xygoto;
[D92]   47:     read2(ch);
[1H6]   48:     if (ch = #0) then begin
[D92]   49:      read2(ch);
[TW3]   50:      raus := false;
[DL0]   51:     end
[A11]   52:     else
[CP3]   53:      raus := (ch = #27);
[P91]   54:     hoch;
[TU0]   55:    end;
[000]   56:
[1ZB]   57:  procedure hilfdir( var exist : boolean );
[BA4]   58:   var saveAx, c1,
[WW4]   59:         c2, c3, c4 : integer;
[PX5]   60:             bereits : boolean;
[BH1]   61:    begin
[FR4]   62:     exist := false;
[TW3]   63:     raus := false;
[TL5]   64:     bereits := false;
[OV4]   65:     saveAx := regs.al;
[W68]   66:     if lo(saveAx) <> 0 then begin
[S84]   67:      if d >= 25 then
[TX5]   68:       if paging then begin
[SL3]   69:        weiter(raus);
[ZM5]   70:        bereits := true;
[TU0]   71:       end;
[VZ6]   72:      if not raus then begin
[P91]   73:       hoch;
[6X6]   74:       speicher(c1, c2, c3, c4, c5);
[HC7]   75:       speicherFrei := 1.0 * c1 * c3;
[ZXA]   76:       speicherKap := 1.0 * c4 * SpeicherFre
        i;
[N89]   77:       SpeicherFrei := SpeicherFrei * c2;
[OD4]   78:       attribut := 17;
[RLA]   79:       SchreibeSatz('Gesamter Speicher :');
[324]   80:       attribut := 9;
[TD7]   81:       MoveReal(9,0,SpeicherKap);
[OD4]   82:       attribut := 17;
[UY5]   83:       SchreibeSatz(' Byte  ');
[324]   84:       attribut := 9;
[MP5]   85:       MoveInt(4,anzahl);
[OD4]   86:       attribut := 17;
[FC7]   87:       SchreibeSatz(' Datei(en)');
[XR3]   88:       if (d >= 24) then
[2ZA]   89:        if paging and not bereits then begin
[SL3]   90:         weiter(raus);
[ZM5]   91:         bereits := true;
[TU0]   92:        end;
[VZ6]   93:       if not raus then begin
[P91]   94:        hoch;
[EDB]   95:        SchreibeSatz('Verzeichnisgröße  :');
[324]   96:        attribut := 9;
[UA8]   97:        MoveReal(9,0,SpeicherBelegt);
[OD4]   98:        attribut := 17;
[NVA]   99:        SchreibeSatz(' Byte   Freier Speiche
        r :');
[324]  100:        attribut := 9;
[ZP7]  101:        MoveReal(9,0,SpeicherFrei);
[OD4]  102:        attribut := 17;
[WV5]  103:        SchreibeSatz(' Byte');
[9S8]  104:        if (d >= 23) and not bereits then
[LN4]  105:         if paging then
[SL3]  106:          weiter(raus);
[P91]  107:         hoch;
[TU0]  108:       end;
[TU0]  109:      end;
[DL0]  110:     end
[A11]  111:     else
[TX5]  112:      if paging then begin
[SL3]  113:       weiter(raus);
[GU3]  114:       if raus then
[344]  115:        exist := true;
[9D1]  116:       d := 1;
[MC4]  117:       attribut := 15;
[BF5]  118:       regs.ax := saveAx;
[TU0]  119:      end;
[TU0]  120:    end;
[000]  121:
[DX5]  122:  procedure nofiles;
[BH1]  123:    begin
[P91]  124:     hoch;
```

```
[LB4]  125:     attribut := 138;
[7YB]  126:     SchreibeSatz('Keine Datei(en) gefunden
       !');
[H65]  127:     hilfdir(exist);
[TU0]  128:    end;
[000]  129:
[2A8]  130:  procedure param( w : byte );
[NW3]  131:   var h : tstr255;
[BH1]  132:    begin
[VP4]  133:     h := paramStr(w);
[NB6]  134:     if h[1] = '/' then begin
[6C6]  135:      if h[2] in ['p','P'] then
[3E4]  136:       paging := true
[A11]  137:      else
[PV5]  138:      if h[2] in ['1'..'6'] then
[TY6]  139:       optionen := ord(h[2])-48;
[DL0]  140:     end
[A11]  141:     else
[AE2]  142:      gef := true;
[TU0]  143:     end;
[000]  144:
[637]  145:  procedure MM( t : byte );
[BH1]  146:    begin
[RF3]  147:     str(t:2,str2);
[6K3]  148:     if t < 10 then
[2M3]  149:      str2[1] := #48;
[AV5]  150:     SchreibeSatz(str2);
[TU0]  151:    end;
[000]  152:
[BH1]  153:   begin
[HO4]  154:    optionen := 7;
[KAA]  155:    paging := false;    { Seitenweise }
[RKC]  156:    gef := false;       { pfad oder Datei ang
       egeben }
[N25]  157:    c := paramcount;
[FO3]  158:    if c > 4 then
[UD1]  159:     c := 4;
[H84]  160:    for i := 1 to c do
[RH2]  161:     param(i);
[HR4]  162:    if gef then begin
[FE5]  163:     str255 := ParamStr(1);
[CP8]  164:     for i := 1 to length(str255) do
[DY8]  165:      str255[i] := upcase(str255[i]);
[DL0]  166:    end
[A11]  167:    else
[NA2]  168:     str255 := '';
[EFF]  169:    if ( str255[ 0 ] = #0 ) or ( str255[ byt
       e( str255[ 0 ] ) ] = '\' ) or
[C6A]  170:       ( str255[ byte( str255[ 0 ] ) ] = ':'
       ) then
[SY4]  171:     str255 := str255 + '*.*';
[G35]  172:    case optionen of
[ZS1]  173:        1 : c := 5;
[Y52]  174:     2, 3 : c := 3;
[362]  175:     4, 5 : c := 2;
[862]  176:     6, 7 : c := 1;
[TU0]  177:    end;
[T7K]  178:    b1 := (optionen in [2,5..7]); {Zugriff ü
       ber Booleans schneller}
[K56]  179:    b2 := (optionen in [3..7]);
[XE6]  180:    b3 := (optionen in [4,6,7]);
[1D1]  181:    b := 0;
[SW2]  182:    anzahl := 0;
[Y66]  183:    speicherBelegt := 0.0;
[9D1]  184:    d := 1;
[G13]  185:    y := wherey;
[CB4]  186:    if y >= 25 then
[P91]  187:     hoch;
[652]  188:    adr(1,y);
[X94]  189:    attribut := 112;
[Y86]  190:    SchreibeSatz('Pfad: ');
[375]  191:    with regs do begin
[NG1]  192:     ah := $19;
[W93]  193:     msdos(regs);
[6P2]  194:     c5 := al + 65;
[KD6]  195:     if str255[2] = ':' then
[AA7]  196:      c5 := ord(upcase(str255[1]));
[PD2]  197:     i := c5 - 64;
[353]  198:     Sicherung;
[594]  199:     aktdir(str80,i);
[934]  200:     EntSicherung;
[ER1]  201:     INT24F;
[584]  202:     if BREAK24H then
[7C1]  203:      exit;
[DP1]  204:     adr(3,4);
[YM4]  205:     str9 := chr(c5) + ':';
[5B6]  206:     if str255[2] <> ':' then
[HV5]  207:      SchreibeSatz(str9);
[SHE]  208:     if (str255[1] <> '\') and (str255[3] <>
       '\') then begin
[8D6]  209:      SchreibeZeichen('\');
[3F6]  210:      if (str80[0] <> #0) then
[ZX6]  211:       SchreibeSatz(str80+'\');
[TU0]  212:     end;
```

```
[V96]  213:      SchreibeSatz(str255);
[RLA]  214:      xygoto;                (Position sichern)
[LP4]  215:      str255 := str255 + #0;
[TS4]  216:      str9 := str9 + '\*.*' + #0;
[XF1]  217:      Ah := $1A;
[1Y3]  218:      DS := Seg(as[1]);
[VX3]  219:      DX := Ofs(as[1]);
[W33]  220:      MSDos(regs);
[1G1]  221:      Ah := $4E;
[VD1]  222:      CX := 9;
[ND4]  223:      DS := Seg(str9[1]);
[FD4]  224:      DX := Ofs(str9[1]);
[WW2]  225:      MSDos(Regs);
[EW1]  226:      Int24F;
[5U4]  227:      if Break24H then
[7C1]  228:       exit;
[CU3]  229:      error := al <> 0;
[P91]  230:      hoch;
[T14]  231:      attribut := 7;
[CPA]  232:      SchreibeSatz('Der Datenträger hat ');
[EM7]  233:      if not error then begin
[MA8]  234:       SchreibeSatz('den Namen : ');
[9D1]  235:       d := 1;
[MC4]  236:       attribut := 15;
[MG6]  237:       while (as[d] <> 0) do begin
[WP8]  238:        SchreibeZeichen(chr(as[d]));
[N33]  239:        d := succ(d);
[TUO]  240:       end;
[DLO]  241:      end
[YS2]  242:      else begin
[MC4]  243:       attribut := 15;
[LA8]  244:       SchreibeSatz('keinen Namen !')
[TUO]  245:      end;
[1G1]  246:      Ah := $4E;
[CE1]  247:      CX := 30;
[OP4]  248:      DS := Seg(str255[1]);
[UN4]  249:      DX := Ofs(str255[1]);
[WW2]  250:      MSDos(Regs);
[EW1]  251:      Int24F;
[5U4]  252:      if Break24H then
[7C1]  253:       exit;
[CU3]  254:      error := al <> 0;
[DX4]  255:      if not(error) then
[3U8]  256:      if boolean(as[22] and 16) then
[D6E]  257:      if (str255[pred(byte(str255[0]))] <> '*
       ') then begin
[4Y9]  258:       delete( str255, length( str255 ), 1 );
       {#0}
[UU5]  259:       adr(wherex,wherey);
[RF5]  260:       SchreibeSatz( '\*.*');
[8V5]  261:       str255 := str255 + '\*.*'+#0;
[1G1]  262:       Ah := $4E;
[CE1]  263:       CX := 30;
[OP4]  264:       DS := Seg(str255[1]);
[UN4]  265:       DX := Ofs(str255[1]);
[WW2]  266:       MSDos(Regs);
[EW1]  267:       Int24F;
[5U4]  268:       if Break24H then
[7C1]  269:        exit;
[TUO]  270:      end;
[TUO]  271:     end;
[P91]  272:     hoch;
[P91]  273:     hoch;
[R17]  274:     while regs.al = 0 do begin
[DV9]  275:      if boolean(as[22] and 8) then begin
[DLO]  276:      end
[YS2]  277:      else begin
[MC4]  278:       attribut := 15;
[3U8]  279:       if boolean(as[22] and 16) then
[X94]  280:        attribut := 112
[A11]  281:       else
[MLB]  282:        if boolean(as[22] and 2) then {Hidden
       }
[T14]  283:         attribut := 7;
[4L6]  284:       SchreibeZeichen('||');
[S86]  285:       move(as[31],zw1[1],13);
[GV2]  286:       zw1[0] := #13;
[1U5]  287:       t := pred(pos(#0,zw1));
[W94]  288:       zw1[0] := chr(t);
[UK4]  289:       if zw1[1] = '.' then
[UK4]  290:        if zw1[2] = '.' then
[R38]  291:         SchreibeSatz('<Übergeord.>')
[A11]  292:        else
[728]  293:         SchreibeSatz('<Aktuelles >')
[YS2]  294:       else begin
[266]  295:        fillchar(zw[1],12,#32);
[8E4]  296:        i := pos('.',zw1);
[KZ4]  297:        if i <> 0 then begin
[NX7]  298:         move(zw1[1],zw[1],pred(i));
[7V8]  299:         move(zw1[i],zw[9],t-pred(i));
[DLO]  300:        end
[A11]  301:        else
[F96]  302:         move(zw1[1],zw[1],t);
```

```
[8K2]  303:        zw[0] := #12;
[FN5]  304:        SchreibeSatz(zw);
[TUO]  305:       end;
[PV6]  306:       anzahl := succ(anzahl);
[92A]  307:       if boolean(as[22] and 16) then begin
[Z92]  308:        if b1 then
[CM7]  309:         SchreibeSatz('||  <Dir> ');
[DLO]  310:       end
[YS2]  311:       else begin
[139]  312:        move(as[27],q2,2); (Schneller)
[DG4]  313:        move(as[29],q1,2);
[GN3]  314:        if q2 < 0 then
[YG3]  315:         q1 := succ(q1);
[Y87]  316:        groesse := 6.5536e4 * q1 + q2;
[9L4]  317:        if b1 then begin
[4L6]  318:         SchreibeZeichen('||');
[7B6]  319:         MoveReal(8,0,groesse);
[TUO]  320:        end;
[6LD]  321:        speicherBelegt := speicherBelegt + gr
       oesse;
[TUO]  322:       end;
[EL4]  323:       if b2 then begin
[4L6]  324:        SchreibeZeichen('||');
[PE8]  325:        t := as[25] and 31;              {Tag
       }
[YA1]  326:        MM(t);
[CA6]  327:        SchreibeZeichen('.');
[T8D]  328:        t := (as[26] and 1) shl 3 + as[25] sh
       r 5;     {Monat}
[YA1]  329:        MM(t);
[CA6]  330:        SchreibeZeichen('.');
[TM8]  331:        i := as[26] shr 1;               {Jahr
       }
[TA4]  332:        str(i+1980,str9);
[HV5]  333:        SchreibeSatz(str9);
[TUO]  334:       end;
[LL4]  335:       if b3 then begin
[4L6]  336:        SchreibeZeichen('||');
[U99]  337:        t := as[24] shr 3;               {Stun
       de}
[YA1]  338:        MM(t);
[4B6]  339:        SchreibeZeichen(':');
[SFD]  340:        t := (as[24] and 7) shl 3 + as[23] sh
       r 5; {Minute}
[4B6]  341:        SchreibeZeichen(':');
[YA1]  342:        MM(t);
[MVA]  343:        t := (as[23] and 31) shl 1;
              {Sekunde}
[YA1]  344:        MM(t);
[TUO]  345:       end;
[B57]  346:       if optionen = 7 then begin
[4L6]  347:        SchreibeZeichen('||');
[2G2]  348:        MM(as[22]);
[TUO]  349:       end;
[CZ6]  350:       if anzahl = 0 then begin
[1Y2]  351:        nofiles;
[7C1]  352:        exit;
[TUO]  353:       end;
[D33]  354:       b := succ(b);
[4L6]  355:       SchreibeZeichen('||');
[T14]  356:       attribut := 7;
[G96]  357:       SchreibeZeichen(' ');
[PF4]  358:       if b = c then begin
[N33]  359:        d := succ(d);
[1D1]  360:        b := 0;
[P91]  361:        hoch;
[TUO]  362:       end;
[TUO]  363:      end;
[A14]  364:      regs.ah := $4F;
[W33]  365:      MSDos(regs);
[EW1]  366:      Int24F;
[5U4]  367:      if Break24H then
[7C1]  368:       exit;
[GSB]  369:      if ( (d = 25) and paging ) or (regs.al
       <> 0) then
[FZ6]  370:       if anzahl > 0 then begin
[H65]  371:        hilfdir(exist);
[DC9]  372:        if exist or (regs.ah <> 0) then
[7C1]  373:         exit;
[TUO]  374:       end;
[TUO]  375:     end;
[2P4]  376:     if anzahl = 0 then
[1Y2]  377:      nofiles;
[TUO]  378:   end;
[000]  379:
[AFC]  380:  begin   {CBreak:=false bringt hier nichts}
[FTA]  381:   ctrlbreak(true,ctrlbreakstatus);
[MP7]  382:   ctrlbreak(false,trueB);
[DK2]  383:   screen;
[WZ1]  384:   int24S;
[SVO]  385:   dir;
[PZ1]  386:   int24R;
[99B]  387:   CtrlBreak(false,CtrlBreakStatus);
[5TO]  388:  end.
```

Listing 13. Quellcode für Directory-Programm »Dir4«

```
[375]    0: {$I-,K-,D-,R-,V-,C-,U-}
[000]    1:
[NAF]    2: {DIR4.PAS 28.8.87, (Geändert für DOS intern
           ational 8.1.88)}
[000]    3:
[WP3]    4: {$I LIBRARY.INC}
[R64]    5: {$I WINDOW.INC}
[XU3]    6: {$I LIBRARYE.INC}
[4P3]    7: {$I TOOLS.INC}
[000]    8:
[2P4]    9: procedure dir;
[BE9]   10:  const         optionen : integer = 7;
[V4C]   11:  type                        ast = array[1
           ..43] of byte;
[LD5]   12:                           tstr12 = string[
           12];
[FS6]   13:  var                         ch : Char;
[5R3]   14:                    anzahl, y, q1,
[D17]   15:                  q2, anzxss, i, c : integer
           ;
[W99]   16:      speicherBelegt, speicherFrei,
[Z88]   17:               groesse, speicherKap : real;
[N62]   18:                              as : ast;
[B54]   19:                           str9 : string[
           9];
[Y24]   20:                           str2 : string[
           2];
[5Z3]   21:                         str255 : tstr255
           ;
[HG5]   22:                 t, c5, a, b, d, e : byte;
[OX5]   23:          paging, gef, b1, b2, b3,
[XV7]   24:                exist, error, raus : boolean
           ;
[AT3]   25:                           str80 : tstr80;
[3W2]   26:                              zw : tstr12;
[T14]   27:                             zw1 : string[
           13];
[X33]   28:                            regs : tregs;
[000]   29:
[6A5]   30:  procedure hoch;
[BH1]   31:    begin
[RW5]   32:     if y >= 25 then begin
[SCC]   33:      rollen1; {Schneller als BIOS-Routine !
           }
[UC1]   34:      y := 25;
[DLO]   35:     end
[A11]   36:     else
[963]   37:      y := succ(y);
[652]   38:     adr(1,y);
[TUO]   39:    end;
[000]   40:
[TOC]   41:  procedure weiter( var raus : boolean );
[5L3]   42:   var    ch : char;
[BH1]   43:    begin
[MC4]   44:     attribut := 15;
[75M]   45:     SchreibeSatz('Für weiter eine Taste bet
           ätigten ! (<ESC> Abbrechen)');
[PP2]   46:     xygoto;
[D92]   47:     read2(ch);
[1H6]   48:     if (ch = #0) then begin
[D92]   49:      read2(ch);
[TW3]   50:      raus := false;
[DLO]   51:     end
[A11]   52:     else
[CP3]   53:      raus := (ch = #27);
[P91]   54:     hoch;
[TUO]   55:    end;
[000]   56:
[1ZB]   57:  procedure hilfdir( var exist : boolean );
[BA4]   58:   var saveAx, c1,
[WW4]   59:         c2, c3, c4 : integer;
[PX5]   60:            bereits : boolean;
[BH1]   61:    begin
[FR4]   62:     exist := false;
[TW3]   63:     raus := false;
[TL5]   64:     bereits := false;
[OV4]   65:     saveAx := regs.al;
[W68]   66:     if lo(saveAx) <> 0 then begin
[S84]   67:      if d >= 25 then
[TX5]   68:       if paging then begin
[SL3]   69:        weiter(raus);
[ZM5]   70:        bereits := true;
[TUO]   71:       end;
[VZ6]   72:      if not raus then begin
[P91]   73:       hoch;
[6X6]   74:       speicher(c1, c2, c3, c4, c5);
[HC7]   75:       speicherFrei := 1.0 * c1 * c3;
[ZXA]   76:       speicherKap := 1.0 * c4 * SpeicherFre
           i;
[N89]   77:       SpeicherFrei := SpeicherFrei * c2;
[OD4]   78:       attribut := 17;
[RLA]   79:       SchreibeSatz('Gesamter Speicher :');
[324]   80:       attribut := 9;
[TD7]   81:       MoveReal(9.0,SpeicherKap);
[OD4]   82:       attribut := 17;
[UY5]   83:       SchreibeSatz(' Byte  ');
[324]   84:       attribut := 9;
[MP5]   85:       MoveInt(4,anzahl);
[OD4]   86:       attribut := 17;
```

```
[FC7]   87:       SchreibeSatz(' Datei(en)');
[XR3]   88:       if (d >= 24) then
[2ZA]   89:        if paging and not bereits then begin
[SL3]   90:         weiter(raus);
[ZM5]   91:         bereits := true;
[TUO]   92:        end;
[VZ6]   93:       if not raus then begin
[P91]   94:        hoch;
[EDB]   95:        SchreibeSatz('Verzeichnisgröße  :');
[324]   96:        attribut := 9;
[UA8]   97:        MoveReal(9,0,SpeicherBelegt);
[OD4]   98:        attribut := 17;
[NVA]   99:        SchreibeSatz(' Byte   Freier Speiche
           r :');
[324]  100:        attribut := 9;
[ZP7]  101:        MoveReal(9,0,SpeicherFrei);
[OD4]  102:        attribut := 17;
[WV5]  103:        SchreibeSatz(' Byte');
[9S8]  104:        if (d >= 23) and not bereits then
[LN4]  105:         if paging then
[SL3]  106:          weiter(raus);
[P91]  107:        hoch;
[TUO]  108:       end;
[TUO]  109:      end;
[DLO]  110:     end
[A11]  111:     else
[TX5]  112:      if paging then begin
[SL3]  113:       weiter(raus);
[GU3]  114:       if raus then
[344]  115:        exist := true;
[9D1]  116:       d := 1;
[MC4]  117:       attribut := 15;
[BF5]  118:       regs.ax := saveAx;
[TUO]  119:      end;
[TUO]  120:    end;
[000]  121:
[DX5]  122:  procedure nofiles;
[BH1]  123:    begin
[P91]  124:     hoch;
[LB4]  125:     attribut := 138;
[7YB]  126:     SchreibeSatz('Keine Datei(en) gefunden
           !');
[H65]  127:     hilfdir(exist);
[TUO]  128:    end;
[000]  129:
[2A8]  130:  procedure param( w : byte );
[NW3]  131:   var h : tstr255;
[BH1]  132:    begin
[VP4]  133:     h := paramStr(w);
[NB6]  134:     if h[1] = '/' then begin
[6C6]  135:      if h[2] in ['p','P'] then
[3E4]  136:       paging := true
[A11]  137:      else
[PV5]  138:      if h[2] in ['1'..'6'] then
[TY6]  139:       optionen := ord(h[2])-48;
[DLO]  140:     end
[A11]  141:     else
[AE2]  142:      gef := true;
[TUO]  143:     end;
[000]  144:
[637]  145:  procedure MM( t : byte );
[BH1]  146:    begin
[RF3]  147:     str(t:2,str2);
[6K3]  148:     if t < 10 then
[2M3]  149:      str2[1] := #48;
[AV5]  150:     SchreibeSatz(str2);
[TUO]  151:    end;
[000]  152:
[BH1]  153:   begin
[HO4]  154:    optionen := 7;
[KAA]  155:    paging := false;    { Seitenweise }
[RKC]  156:    gef := false;       { pfad oder Datei ang
           egeben }
[N25]  157:    c := paramcount;
[FO3]  158:    if c > 4 then
[UD1]  159:     c := 4;
[H84]  160:    for i := 1 to c do
[RH2]  161:     param(i);
[HR4]  162:    if gef then begin
[FE5]  163:     str255 := ParamStr(1);
[CP8]  164:     for i := 1 to length(str255) do
[DY8]  165:      str255[i] := upcase(str255[i]);
[DLO]  166:    end
[A11]  167:    else
[NA2]  168:     str255 := '';
[EFF]  169:    if ( str255[ 0 ] = #0 ) or ( str255[ byt
           e( str255[ 0 ] ) ] = '\' ) or
[C6A]  170:        ( str255[ byte( str255[ 0 ] ) ] = ':'
           ) then
[SY4]  171:     str255 := str255 + '*.*';
[G35]  172:    case optionen of
[ZS1]  173:       1 : c := 5;
[Y52]  174:     2, 3 : c := 3;
[362]  175:     4, 5 : c := 2;
[862]  176:     6, 7 : c := 1;
[TUO]  177:    end;
[T7K]  178:    b1 := (optionen in [2,5..7]); {Zugriff ü
           ber Booleans schneller}
[K56]  179:    b2 := (optionen in [3..7]);
```

```
[XE6]  180:     b3 := (optionen in [4,6,7]);
[1D1]  181:     b := 0;
[SW2]  182:     anzahl := 0;
[Y66]  183:     speicherBelegt := 0.0;
[9D1]  184:     d := 1;
[G13]  185:     y := wherey;
[CB4]  186:     if y >= 25 then
[P91]  187:      hoch;
[652]  188:     adr(1,y);
[X94]  189:     attribut := 112;
[Y86]  190:     SchreibeSatz('Pfad: ');
[375]  191:     with regs do begin
[NG1]  192:      ah := $19;
[W93]  193:      msdos(regs);
[6P2]  194:      c5 := al + 65;
[KD6]  195:      if str255[2] = ':' then
[AA7]  196:       c5 := ord(upcase(str255[1]));
[PD2]  197:      i := c5 - 64;
[353]  198:      Sicherung:
[594]  199:      aktdir(str80,i);
[934]  200:      EntSicherung:
[ER1]  201:      INT24F;
[584]  202:      if BREAK24H then
[7C1]  203:       exit;
[DP1]  204:      adr(3,4);
[YM4]  205:      str9 := chr(c5) + ':';
[5B6]  206:      if str255[2] <> ':' then
[HV5]  207:       SchreibeSatz(str9);
[SHE]  208:      if (str255[1] <> '\') and (str255[3] <>
       '\') then begin
[8D6]  209:       SchreibeZeichen('\');
[3F6]  210:       if (str80[0] <> #0) then
[ZX6]  211:        SchreibeSatz(str80+'\');
[TUO]  212:      end;
[V96]  213:      SchreibeSatz(str255);
[RLA]  214:      xygoto;                 {Position sichern}
[LP4]  215:      str255 := str255 + #0;
[TS4]  216:      str9 := str9 + '\*.*' + #0;
[XF1]  217:      Ah := $1A;
[1Y3]  218:      DS := Seg(as[1]);
[VX3]  219:      DX := Ofs(as[1]);
[W33]  220:      MSDos(regs);
[1G1]  221:      Ah := $4E;
[VD1]  222:      CX := 9;
[ND4]  223:      DS := Seg(str9[1]);
[FD4]  224:      DX := Ofs(str9[1]);
[WW2]  225:      MSDos(Regs);
[EW1]  226:      Int24F;
[5U4]  227:      if Break24H then
[7C1]  228:       exit;
[CU3]  229:      error := al <> 0;
[P91]  230:      hoch;
[T14]  231:      attribut := 7;
[CPA]  232:      SchreibeSatz('Der Datentrager hat ');
[EM7]  233:      if not error then begin
[MA8]  234:       SchreibeSatz('den Namen : ');
[9D1]  235:       d := 1;
[MC4]  236:       attribut := 15;
[MG6]  237:       while (as[d] <> 0) do begin
[WP8]  238:        SchreibeZeichen(chr(as[d]));
[N33]  239:        d := succ(d);
[TUO]  240:       end;
[DLO]  241:      end
[YS2]  242:      else begin
[MC4]  243:       attribut := 15;
[LA8]  244:       SchreibeSatz('keinen Namen !')
[TUO]  245:      end;
[1G1]  246:      Ah := $4E;
[CE1]  247:      CX := 30;
[OP4]  248:      DS := Seg(str255[1]);
[UN4]  249:      DX := Ofs(str255[1]);
[WW2]  250:      MSDos(Regs);
[EW1]  251:      Int24F;
[5U4]  252:      if Break24H then
[7C1]  253:       exit;
[CU3]  254:      error := al <> 0;
[DX4]  255:      if not(error) then
[3U8]  256:      if boolean(as[22] and 16) then
[D6E]  257:      if (str255[pred(byte(str255[0]))] <> '*
       ') then begin
[4Y9]  258:       delete( str255, length( str255 ), 1 );
       {#0}
[UU5]  259:       adr(wherex,wherey);
[RF5]  260:       SchreibeSatz( '\*.*');
[8V5]  261:       str255 := str255 + '\*.*'+#0;
[1G1]  262:       Ah := $4E;
[CE1]  263:       CX := 30;
[OP4]  264:       DS := Seg(str255[1]);
[UN4]  265:       DX := Ofs(str255[1]);
[WW2]  266:       MSDos(Regs);
[EW1]  267:       Int24F;
[5U4]  268:       if Break24H then
[7C1]  269:        exit;
[TUO]  270:      end;
[TUO]  271:     end;
[P91]  272:     hoch;
[P91]  273:     hoch;
[R17]  274:     while regs.al = 0 do begin
[DV9]  275:      if boolean(as[22] and 8) then begin
[DLO]  276:      end
[YS2]  277:      else begin
[MC4]  278:       attribut := 15;
[3U8]  279:       if boolean(as[22] and 16) then
[X94]  280:        attribut := 112
[A11]  281:       else
[MLB]  282:        if boolean(as[22] and 2) then {Hidden
       }
[T14]  283:         attribut := 7;
[4L6]  284:       SchreibeZeichen('|');
[S86]  285:       move(as[31],zw1[1],13);
[GV2]  286:       zw1[0] := #13;
[1U5]  287:       t := pred(pos(#0,zw1));
[W94]  288:       zw1[0] := chr(t);
[UK4]  289:       if zw1[1] = '.' then
[UK4]  290:        if zw1[2] = '.' then
[R38]  291:         SchreibeSatz('<Ubergeord.>')
[A11]  292:        else
[728]  293:         SchreibeSatz('<Aktuelles >')
[YS2]  294:       else begin
[266]  295:        fillchar(zw[1],12,#32);
[8E4]  296:        i := pos('.',zw1);
[KZ4]  297:        if i <> 0 then begin
[NX7]  298:         move(zw1[1],zw[1],pred(i));
[7V8]  299:         move(zw1[i],zw[9],t-pred(i));
[DLO]  300:        end
[A11]  301:        else
[F96]  302:         move(zw1[1],zw[1],t);
[8K2]  303:        zw[0] := #12;
[FN5]  304:        SchreibeSatz(zw);
[TUO]  305:       end;
[PV6]  306:       anzahl := succ(anzahl);
[92A]  307:       if boolean(as[22] and 16) then begin
[Z92]  308:        if b1 then
[CM7]  309:         SchreibeSatz('|  <Dir> ');
[DLO]  310:       end
[YS2]  311:       else begin
[139]  312:        move(as[27],q2,2); {Schneller}
[DG4]  313:        move(as[29],q1,2);
[GN3]  314:        if q2 < 0 then
[YG3]  315:         q1 := succ(q1);
[Y87]  316:        groesse := 6.5536e4 * q1 + q2;
[9L4]  317:        if b1 then begin
[4L6]  318:         SchreibeZeichen('|');
[7B6]  319:         MoveReal(8,0,groesse);
[TUO]  320:        end;
[6LD]  321:        speicherBelegt := speicherBelegt + gr
       oesse;
[TUO]  322:       end;
[EL4]  323:       if b2 then begin
[4L6]  324:        SchreibeZeichen('|');
[PE8]  325:        t := as[25] and 31;              {Tag
       }
[YA1]  326:        MM(t);
[CA6]  327:        SchreibeZeichen('.');
[T8D]  328:        t := (as[26] and 1) shl 3 + as[25] sh
       r 5;     {Monat}
[YA1]  329:        MM(t);
[CA6]  330:        SchreibeZeichen('.');
[TM8]  331:        i := as[26] shr 1;               {Jahr
       }
[TA4]  332:        str(i+1980,str9);
[HV5]  333:        SchreibeSatz(str9);
[TUO]  334:       end;
[LL4]  335:       if b3 then begin
[4L6]  336:        SchreibeZeichen('|');
[U99]  337:        t := as[24] shr 3;               {Stun
       de}
[YA1]  338:        MM(t);
[4B6]  339:        SchreibeZeichen(':');
[SFD]  340:        t := (as[24] and 7) shl 3 + as[23] sh
       r 5; {Minute}
[4B6]  341:        SchreibeZeichen(':');
[YA1]  342:        MM(t);
[MVA]  343:        t := (as[23] and 31) shl 1;
          {Sekunde}
[YA1]  344:        MM(t);
[TUO]  345:       end;
[B57]  346:       if optionen = 7 then begin
[4L6]  347:        SchreibeZeichen('|');
[2G2]  348:        MM(as[22]);
[TUO]  349:       end;
[CZ6]  350:       if anzahl = 0 then begin
[1Y2]  351:        nofiles;
[7C1]  352:        exit;
[TUO]  353:       end;
[D33]  354:       b := succ(b);
[4L6]  355:       SchreibeZeichen('|');
[T14]  356:       attribut := 7;
[G96]  357:       SchreibeZeichen(' ');
[PF4]  358:       if b = c then begin
[N33]  359:        d := succ(d);
[1D1]  360:        b := 0;
[P91]  361:        hoch;
[TUO]  362:       end;
[TUO]  363:      end;
[A14]  364:      regs.ah := $4F;
[W33]  365:      MSDos(regs);
[EW1]  366:      Int24F;
```

```
[5U4]  367:      if Break24H then
[7C1]  368:       exit;
[GSB]  369:      if ( (d = 25) and paging ) or (regs.al
         <> 0) then
[FZ6]  370:       if anzahl > 0 then begin
[H65]  371:        hilfdir(exist);
[DC9]  372:        if exist or (regs.ah <> 0) then
[7C1]  373:         exit;
[TU0]  374:       end;
[TU0]  375:     end;
[2P4]  376:     if anzahl = 0 then
[1Y2]  377:      nofiles;
[TU0]  378:    end;
[000]  379:
[AFC]  380:  begin   {CBreak:=false bringt hier nichts}
[FTA]  381:   ctrlbreak(true,ctrlbreakstatus);
[MP7]  382:   ctrlbreak(false,trueB);
[DK2]  383:   screen;
[WZ1]  384:   int24S;
[SV0]  385:   dir;
[PZ1]  386:   int24R;
[99B]  387:   CtrlBreak(false,CtrlBreakStatus);
[5T0]  388:  end.
```

Listing 14. Quellcode für Directory-Programm »Dir6«

```
[P65]    0: {$I-,V-,C-,U-,K-,D-,R-}
[000]    1:
[C7G]    2: {SDIR.PAS 11/12.11.87 (Geändert für DOS int
           ernational 8.1.88)}
[000]    3:
[WP3]    4: {$I LIBRARY.INC}
[R64]    5: {$I WINDOW.INC}
[XU3]    6: {$I LIBRARYE.INC}
[SY8]    7: Type TSetOfChar = Set of Char;
[WP3]    8: {$I READSTR.INC}
[4P3]    9: {$I TOOLS.INC}
[000]   10:
[6SH]   11: const    maxDateiAnzahl = 512;       {Stat.
           Speicherstruktur}
[DB6]   12:                          ab : byte = 1;  {Filte
           r}
[TG4]   13:                         bis : byte = 13; {   "
           }
[ZG8]   14: Type              tstr12 = string[12];
[P15]   15:                    tstr8 = string[8];
[G65]   16:                   tdatei = record
[S53]   17:                                    name : tstr1
           2;
[0X2]   18:                                 passend,
[PF2]   19:                                       a : byte;
[4C2]   20:                                      gr : real;
[5Y3]   21:                                   datum : tstr8
           ;
[TU0]   22:                                 end;
[4G7]   23: var                  regs : tregs;
[FV4]   24:        Seite, MaxSeite,
[1S0]   25:                      Ind,
[366]   26:      a, s, m, st, t, mo : byte;
[A23]   27:                PassendGr,
[G15]   28:           Frei, Kap, gr : real;
[ON6]   29:         pfad, savePfad : tstr80;
[OW2]   30:                 Passend,
[YZ8]   31:  DateiAnzahl, q1, q2, j : integer;
[BP4]   32:       BSort, auswahlB,
[HD5]   33:           BF, BB, n, f : boolean;
[Z14]   34:          saveLaufwerk,
[8H5]   35:               Laufwerk : byte;
[8X2]   36:              Referenz,
[6S5]   37:               altdatei : tstr12;
[AD5]   38:                altpfad : tstr80;
[03D]   39:                  Datei : array[1..maxDatei
           Anzahl] of tdatei;
[000]   40:
[OBK]   41: procedure chdirectory(datei : tstr80; var f
           ehler : boolean );
[Z65]   42:  var regs : tregs;
[BH1]   43:   begin
[5W4]   44:    datei := datei + #0;
[D64]   45:    regs.ah := $3b;
[A97]   46:    regs.dx := ofs(datei[1]);
[A67]   47:    regs.ds := seg(datei[1]);
[W93]   48:    msdos(regs);
[EW1]   49:    Int24F;
[NE8]   50:    fehler := ( regs.flags and 1 ) = 1;
[TU0]   51:   end;
[000]   52:
[CB5]   53: procedure wahl;
[3S5]   54:  var quit : boolean;
[000]   55:
[P44]   56:  procedure sp;
[T4D]   57:   const  st0 : string[21] = 'Gesamter Speic
           her  : ';
[E36]   58:          st1 : string[05] = ' Byte';
```

```
[XRA]   59:          st3 : string[21] = 'Belegter Speic
           her  : ';
[Y8A]   60:          st4 : string[21] = 'Freier Speiche
           r    : ';
[E0B]   61:          st2 : string[21] = 'Anzahl der Dat
           eien : ';
[ZYA]   62:          st7 : string[19] = 'Gefilterte Dat
           eien ';
[BDA]   63:          st6 : string[21] = 'Such-Kriterium
           : ';
[8C9]   64:          stA : string[21] = 'Sortieren
           : ';
[BT8]   65:          stB : string[10] = 'geschaltet';
[WG7]   66:    var     c1, c2, c3, c4 : integer;
[ZA4]   67:                       st9 : string[9];
[VW2]   68:                       stH : tstr12;
[Y53]   69:                         f : boolean;
[EG2]   70:                         y : byte;
[D92]   71:                        ch : char;
[K78]   72:                         a : array[10..19,1.
           .74] of byte;
[BH1]   73:    begin
[1F2]   74:     f := false;
[KT9]   75:     speicher(c1, c2, c3, c4, LaufWerk+64);
[9K4]   76:     Frei := 1.0 * c1 * c3;
[6E3]   77:     Kap := Frei * c4;
[5P3]   78:     Frei := Frei * c2;
[E99]   79:     MakeWindow( 2, 112, 112, 32, 14, 10, 37
           , 10, a );
[X94]   80:     attribut := 112;
[8A3]   81:     SetAdr(15,11);
[PNH]   82:     SchreibeSatz(st0); MoveReal(9,0,Kap);
                    SchreibeSatz(st1);
[ZV2]   83:     NextLine;
[BSK]   84:     SchreibeSatz(st3); MoveReal(9,0,Kap-Fre
           i);      SchreibeSatz(st1);
[ZV2]   85:     NextLine;
[UZH]   86:     SchreibeSatz(st4); MoveReal(9,0,Frei);
                    SchreibeSatz(st1);
[ZV2]   87:     NextLine;
[SXC]   88:     SchreibeSatz(st2); MoveInt(9,DateiAnzah
           l);
[ZV2]   89:     NextLine;
[ONC]   90:     SchreibeSatz(st7+': '); MoveInt(9,Passe
           nd);
[ZV2]   91:     NextLine;
[CWL]   92:     SchreibeSatz(st7+'in'); MoveReal(9,0,Pa
           ssendgr); SchreibeSatz(st1);
[ZV2]   93:     NextLine;
[X9G]   94:     SchreibeSatz(st6+' '+Referenz); {Refere
           nz[0] immer 12}
[ZV2]   95:     NextLine;
[EZ3]   96:     if BSort then
[6U8]   97:      SchreibeSatz(stA+' '+'Ein'+stB)
[A11]   98:     else
[FT8]   99:      SchreibeSatz(stA+' '+'Aus'+stB);
[U78]  100:     NextLine; MemAdr := MemAdr+2;
[809]  101:     SchreibeSatz('Taste drücken');
[CL2]  102:     clear(ch);
[5L7]  103:     RestoreWindow( 14, 10, 37, 10, a );
[TU0]  104:    end;
[000]  105:
[BM8]  106:   procedure NeuesLaufwerk;
[32K]  107:    const st0 : string[33] = 'Neues Laufwerk
          (<ESC> Abbrechen) :';
[BV7]  108:    var     h : array[1..160] of byte;
[D92]  109:           ch : char;
[6D2]  110:           dr : byte;
[BH1]  111:     begin
[VT8]  112:      Move(mem[screenaddr:3360],h,160);
[SB4]  113:      attribut := 112;
[P65]  114:      MoveStr(1,22,st0);
[NU5]  115:      gotoxy(34,22); clreol;
[G93]  116:      Cursor(1,13);
[XN3]  117:      gotoxy(35,22);
[GC8]  118:      MoveChr(35,22,chr(LaufWerk+64));
[X52]  119:      repeat
[D92]  120:       read2(ch);
[YE4]  121:       ch := upcase(ch);
[UY5]  122:       if ch = #0 then begin
[D92]  123:        read2(ch);
[9G1]  124:        ch := #32;
[TU0]  125:       end;
[M77]  126:      until ch in [#13,#27,'A'..'Z'];
[SL6]  127:      if ch <> #27 then begin
[CC4]  128:       attribut := 32;
[R04]  129:       window(1,1,80,22);
[S43]  130:       gotoxy(1,1);
[ORH]  131:  {Falls DOS-Meldung "Diskette in Laufwerk..
           ." (Bei HD + 1 DD) kommt.
[M8N]  132:   Oder: Verwendung der in der LIBRARY entha
           ltenen Proc's SICHERN,ENTSICHERN}
[NM2]  133:       clrscr;
[954]  134:       if ch = #13 then
[BT3]  135:        dr := LaufWerk
[A11]  136:       else
[P93]  137:        dr := ord(ch);
[8W3]  138:       NewDrive(dr);
```

```
[P05]  139:        Laufwerk := dr+64;
[1D2]  140:        n := true;
[GM4]  141:        BSort := false;
[1G2]  142:        BB:= false;
[YH2]  143:        BF := false;
[T76]  144:        Referenz := '????????.???';
[TU0]  145:       end;
[SK3]  146:       Cursor(32,32);
[FS8]  147:       Move(h,mem[screenaddr:3360],160);
[TU0]  148:      end;
[000]  149:
[WCG]  150:   function vergleich( Referenz, Target : ts
       tr12 ) : boolean;
[W63]  151:     var i : byte;
[BH1]  152:      begin
[ZY1]  153:       i := ab;
[VFG]  154:       while ((Referenz[i] = Target[i]) or (R
       eferenz[i] = '?')) and
[6V2]  155:             (i < bis) do
[943]  156:        i := succ(i);
[WM5]  157:       vergleich := (i=bis);
[TU0]  158:      end;
[000]  159:
[RMB]  160:   procedure einzeln( Referenz : tstr12 );
[EM0]  161:    var
[R83]  162:        i : integer;
[4C2]  163:          gr : real;
[BH1]  164:     begin
[AD2]  165:      Seite := 1;
[VHA]  166:      if Referenz = '????????.???' then begi
       n
[H84]  167:       Passendgr := 0.0;
[0U7]  168:       Passend := DateiAnzahl;
[DX9]  169:       MaxSeite := succ(Passend div 90);
[459]  170:       for i := 1 to DateiAnzahl do begin
[2R6]  171:        datei[i].passend := 1;
[52B]  172:        Passendgr := Passendgr + datei[i].gr
       ;
[TU0]  173:       end;
[YH2]  174:       BF := false;
[DL0]  175:      end
[YS2]  176:      else begin
[VL3]  177:       Passend := 0;
[H84]  178:       PassendGr := 0.0;
[GD7]  179:       for i := 1 to DateiAnzahl do
[FHL]  180:        if boolean(Datei[i].a and 16) then
       (Passend ist bei Attr.=16 immer 1 !!!)
[G98]  181:         Passend := succ(Passend)
[A11]  182:        else
[NPE]  183:         if vergleich( Referenz, Datei[i].na
       me ) then begin
[2P6]  184:          Datei[i].passend := 1;
[BB8]  185:          Passend := succ(Passend);
[50B]  186:          Passendgr := Passendgr + Datei[i].
       gr;
[DL0]  187:         end
[A11]  188:         else
[XN6]  189:          Datei[i].passend := 0;
[DX9]  190:       MaxSeite := succ(Passend div 90);
[TU0]  191:      end;
[TU0]  192:     end;
[000]  193:
[YR5]  194:  procedure filter;
[SM9]  195:   type tsetofchar = Set of char;
[H56]  196:   var        Hilf : tstr12;
[F04]  197:              t, i : integer;
[GG2]  198:                H8 : tstr8;
[494]  199:                H3 : string[3];
[BK7]  200:                 h : array[1..3,1..60] of by
       te;
[000]  201:
[5Z8]  202:     procedure Fill( var s : tstr12 );
[PT4]  203:      var i, j : integer;
[353]  204:          Hilf : tstr12;
[BH1]  205:       begin
[G14]  206:        j := length(s);
[HP3]  207:        i := pos('*',s);
[113]  208:        if i = 0 then
[7C1]  209:         exit;
[F37]  210:        if pos(#32,s) > 0 then begin
[E59]  211:         if s[succ(i)] <> #32 then begin
[XC6]  212:          fillchar(Hilf[1],13,#32);
[5T4]  213:          Hilf := copy(s,1,i);
[LL4]  214:          Hilf[0] := chr(j);
[65D]  215:          Move(s[succ(i)],Hilf[j-pred(pos(#3
       2,s)-succ(i))],
[GS4]  216:                pos(#32,s)-succ(i));
[032]  217:          s := Hilf;
[TU0]  218:         end;
[ED1]  219:        i := 1;
[M56]  220:        while i <= j do begin
[VB6]  221:         if s[i] in [#32,'*'] then
[5C2]  222:          s[i] := '?';
[943]  223:         i := succ(i);
[TU0]  224:        end;
[DL0]  225:       end
[A11]  226:       else
[6G4]  227:        s[pos('*',s)] := '?';
```

```
[TU0]  228:       end;
[000]  229:
[BH1]  230:       begin
[X94]  231:        attribut := 112;
[X38]  232:        MakeWindow(4,112,112,32,10,10,30,3,h
       );
[0A3]  233:        SetAdr(11,10);
[6CB]  234:        SchreibeSatz('Datei-Filter'); NextLi
       ne;
[4U8]  235:        SchreibeSatz('Wildcards *,?');
[G93]  236:        Cursor(1,13);
[TC4]  237:        attribut := 25;
[X52]  238:        repeat
[9VC]  239:         readStr(Referenz,[#32,#33,#35..#42,
       #45,#46,#48..#58,#63..#90,
[SE6]  240:                          #94..#123,#125..#
       127],12,26,11,true);
[X2E]  241:         if (Referenz = '') or (Referenz[0]
       = #255) then begin
[T76]  242:          Referenz := '????????.???';
[A17]  243:          RestoreWindow(10,10,30,3,h);
[7C1]  244:          exit;
[TU0]  245:         end;
[ED1]  246:         i := 1;
[LD1]  247:         t := 0;
[VMA]  248:         while i <= length(Referenz) do begi
       n
[5V6]  249:          if Referenz[i] = '.' then
[N53]  250:           t := succ(t);
[943]  251:          i := succ(i);
[TU0]  252:         end;
[M74]  253:        until t in [0,1];
[SK3]  254:        Cursor(32,32);
[2D7]  255:        if Referenz <> '' then begin
[W86]  256:         if Referenz = '*.*' then
[826]  257:          Referenz := '????????.???'
[YS2]  258:         else begin
[GA8]  259:          t := succ(length(Referenz));
[UC6]  260:          fillchar(Hilf[1],12,#32);
[6P3]  261:          Hilf[0] := #12;
[076]  262:          i := pos('.',Referenz);
[SG4]  263:          if i > 0 then begin
[SG4]  264:           if i > 9 then begin
[LZ6]  265:            delete(Referenz,1,i-9);
[YE1]  266:            i := 9;
[TU0]  267:           end;
[TT9]  268:           move(Referenz[1],Hilf[1],pred(i))
       ;
[WD4]  269:           if t-i > 4 then
[MP1]  270:            t := i+4;
[F09]  271:           move(Referenz[i],Hilf[9],t-i);
[DL0]  272:          end
[A11]  273:          else
[77D]  274:           Move(Referenz[1],Hilf[1],length(R
       eferenz));
[SG4]  275:           if i > 0 then begin
[PD6]  276:            (pos('.',Referenz) > 0)
[VK5]  277:            H3 := copy(Hilf,10,12);
[412]  278:            Fill(H3);
[VZ4]  279:            H8 := copy(Hilf,1,8);
[722]  280:            Fill(H8);
[AD5]  281:            Referenz := H8+'.'+H3;
[DL0]  282:           end
[YS2]  283:           else begin
[CP2]  284:            Fill(Hilf);
[M25]  285:            Referenz := Hilf;
[TU0]  286:           end;
[TU0]  287:          end;
[ND1]  288:          ab := 1;
[A57]  289:          while Referenz[ab] = '?' do
[5T3]  290:           ab := succ(ab);
[MG6]  291:          Bis := length(Referenz);
[TF7]  292:          while Referenz[Bis] = '?' do
[544]  293:           Bis := pred(Bis);
[144]  294:          Bis := succ(Bis);
[KZ6]  295:          MoveStr(26,11,Referenz);
[706]  296:          einzeln(Referenz);
[TU0]  297:         end;
[A17]  298:         RestoreWindow(10,10,30,3,h);
[TU0]  299:        end;
[000]  300:
[VU5]  301:  procedure sorter;
[000]  302:
[THD]  303:   procedure sort( v, b : integer ); {Quicks
       ort}
[PT4]  304:    var i, j : integer;
[ZU2]  305:          vg : tstr12;
[1S2]  306:           h : tdatei;
[BH1]  307:     begin
[6P7]  308:      vg := datei[(v+b) shr 1].name;
[CU1]  309:      i := v;
[NP1]  310:      j := b;
[X52]  311:      repeat
[EA7]  312:       while datei[i].name < vg do
[943]  313:        i := succ(i);
[XA7]  314:       while datei[j].name > vg do
[733]  315:        j := pred(j);
[515]  316:       if i <= j then begin
```

```
[8S3]  317:          h := datei[i];
[DA6]  318:          datei[i] := datei[j];
[034]  319:          datei[j] := h;
[943]  320:          i := succ(i);
[733]  321:          j := pred(j);
[TU0]  322:        end;
[M53]  323:       until i > j;
[B13]  324:       if j > v then
[PE2]  325:        sort(v,j);
[V03]  326:       if i < b then
[8C2]  327:        sort(i,b);
[TU0]  328:     end;
[000]  329:
[BH1]  330:    begin
[BC5]  331:     sort(1,DateiAnzahl);
[WH2]  332:     signal;
[TU0]  333:    end;
[000]  334:
[VTG]  335:  function Search( x1, y1 : byte; Seite : by
       te) : integer;
[CC5]  336:   var h, i, j : integer;
[BH1]  337:    begin
[DKC]  338:     j := (succ)(pred(Seite) * 90) + (pred(y
       1)*5+x1);
[KK4]  339:     if BF then begin
[ED1]  340:      i := 1;
[7D1]  341:      h := 0;
[X52]  342:      repeat
[WK7]  343:       if Datei[i].Passend = 1 then
[543]  344:        h := succ(h);
[943]  345:       i := succ(i);
[XN2]  346:      until (h=j);
[4S4]  347:      search := pred(i);
[DL0]  348:     end
[A11]  349:     else
[E03]  350:      search := j;
[TU0]  351:    end;
[000]  352:
[6X3]  353: {$I PULLDOW1.INC}
[000]  354:
[RN9]  355:  procedure grz( x1,y1,seite : byte);
[1C4]  356:   var    i : integer;
[B54]  357:       str9 : string[9];
[BH1]  358:    begin
[VU6]  359:     i := search(x1,y1,seite);
[CC4]  360:     attribut := 32;
[MR7]  361:     str(Datei[i].gr:9:0,str9);
[YB5]  362:     MoveStr(8,23,str9);
[DM6]  363:     str(Datei[i].a:3,str9);
[D75]  364:     MoveStr(40,23,str9);
[FA8]  365:     MoveStr(54,23,Datei[i].datum);
[TU0]  366:    end;
[000]  367:
[PZ5]  368:  procedure NextPage;
[BH1]  369:    begin
[9L9]  370:     if Seite < MaxSeite then begin
[1D2]  371:      n := true;
[V82]  372:      BB := true;
[RE5]  373:      Seite := succ(Seite);
[TU0]  374:     end;
[TU0]  375:    end;
[000]  376:
[7Y5]  377:  procedure LastPage;
[BH1]  378:    begin
[FA7]  379:     if Seite > 1 then begin
[1D2]  380:      n := true;
[V82]  381:      BB := true;
[NE5]  382:      Seite := pred(Seite);
[TU0]  383:     end;
[TU0]  384:    end;
[000]  385:
[RL7]  386:  procedure go( ch : byte );
[BH1]  387:    begin
[RS2]  388:     case ch of
[TH9]  389:       80 : if y1 < yanz then     (runter)
[EF0]  390:             g(1)
[A11]  391:            else
[KT2]  392:             NextPage;
[Y47]  393:       72 : if y1 > 1 then        (hoch)
[HF0]  394:             g(2)
[A11]  395:            else
[TS2]  396:             LastPage;
[9B7]  397:       75 : if x1 > 1 then        (links)
[MF0]  398:             g(3)
[A11]  399:            else
[UP3]  400:             if y1 > 1 then
[UF0]  401:              g(5)
[A11]  402:             else
[TS2]  403:              LastPage;
[UA9]  404:       77 : if x1 < xanz then     (rechts)
[RF0]  405:             g(4)
[A11]  406:            else
[XE4]  407:             if y1 < yanz then
[XF0]  408:              g(6)
[A11]  409:             else
[KT2]  410:              NextPage;
[UC5]  411:       81 : NextPage; (Pgdn)
[KB5]  412:       73 : LastPage (Pgup)
```

```
[A11]  413:       else
[LE4]  414:        if ch = 59 then
[4GO]  415:         sp;
[TU0]  416:      end;
[TU0]  417:     end;
[000]  418:
[BH1]  419:     begin
[TW3]  420:      raus := false;
[NN2]  421:      ok := false;
[PH2]  422:      BB := false;
[9F2]  423:      n := false;
[FB7]  424:      invers1( a, x, y, anz, true );
[4U4]  425:      grz(x1,y1,seite);
[X52]  426:      repeat
[862]  427:       read2( ch );
[YM5]  428:       if ( ch = #0 ) then begin
[862]  429:        read2( ch );
[4U2]  430:        go(ord(ch));
[4U4]  431:        grz(x1,y1,seite);
[DL0]  432:       end
[YS2]  433:       else begin
[YE4]  434:        ch := upcase(ch);
[RS2]  435:        case ch of
[RXE]  436:          #13 : ok := boolean(Datei[search(x1
       ,y1,seite)].a and 16);
[Z52]  437:          #27 : begin
[SB3]  438:                raus := true;
[GM4]  439:                BSort := false;
[TU0]  440:               end;
[8A5]  441:          #78 : neuesLaufwerk;
[LE0]  442:          #52,
[TE0]  443:          #54,
[ZE0]  444:          #56,
[F64]  445:          #60 : go(ord(ch));
[M52]  446:          #70 : begin
[EL2]  447:                Filter;
[1D2]  448:                n := true;
[V82]  449:                BB := true;
[392]  450:                BF := true;
[TU0]  451:               end;
[M08]  452:          #83 : if not BSort then begin
[TN2]  453:                sorter;
[1D2]  454:                n := true;
[V82]  455:                BB := true;
[414]  456:                BSort := true;
[DL0]  457:               end
[YS2]  458:               else begin
[GM4]  459:                BSort := false;
[1D2]  460:                n := true;
[V82]  461:                BB := true;
[TU0]  462:               end;
[TU0]  463:        end;
[TU0]  464:       end;
[TS6]  465:      until ok or raus or n;
[MC4]  466:      attribut := 15;
[TU0]  467:     end;
[000]  468:
[ZY5]  469:  procedure auswahl;
[KMB]  470:   const        a1 = 15;     a = 32;      x =
       2; y = 2; mxs = 74;
[617]  471:                mys = 23; mxss = 148; mys1 =
       21;
[9P6]  472:              Leer : string[6] = 'Leer !';
[NM5]  473:   var           i : integer;
[KH4]  474:       xs, ys1, ys : byte;
[0T8]  475:                h1 : array[1..mys1,1..mxss]
       of byte;
[0U9]  476:                 h : array[1..mys,1..mxss] o
       f byte;
[000]  477:
[PB7]  478:   procedure moveDateien;
[4M7]  479:    var i, j, g, xa, ya : integer;
[BH1]  480:     begin
[4R8]  481:      if BF and (MaxSeite > 1) then begin
[ZP6]  482:       j := succ(pred(Seite) * 90);
[ED1]  483:       i := 1;
[6D1]  484:       g := 0;
[X52]  485:       repeat
[WK7]  486:        if Datei[i].Passend = 1 then
[143]  487:         g := succ(g);
[943]  488:        i := succ(i);
[XN2]  489:       until (g=j);
[333]  490:       i := pred(i);
[DL0]  491:      end
[A11]  492:      else
[YP6]  493:       i := succ(pred(Seite) * 90);
[AE1]  494:      xa := 1;
[BE1]  495:      ya := 1;
[6D1]  496:      g := 0;
[7H7]  497:      fillchar( h1,sizeof(h1),32 );
[2K6]  498:      if Passend > 0 then begin
[X52]  499:       repeat
[GF9]  500:        if Datei[i].passend = 1 then begin
[143]  501:         g := succ(g);
[K56]  502:         for j := 1 to 12 do begin
[ZTE]  503:          h1[ya,xa + pred(j) shl 1] := ord(D
       atei[i].name[j]);
[CA9]  504:          if boolean(Datei[i].a and 16) then
```

```
[SZ8]  505:            h1[ya,succ(xa + pred(j) shl 1)] :
       = 15
[TU0]  506:          end;
[M96]  507:          if xa = 121 then begin
[AE1]  508:           xa := 1;
[PW3]  509:           ya := succ(ya);
[DL0]  510:          end
[A11]  511:          else
[FX2]  512:           xa := xa + 30;
[TU0]  513:        end;
[943]  514:        i := succ(i);
[G19]  515:       until (i > DateiAnzahl) or (g = 90);
[ZX3]  516:       if xa = 1 then
[FN4]  517:        ys := ya   {pred}
[A11]  518:       else
[7K4]  519:        ys := succ(ya);  {}
[DL0]  520:      end
[A7L]  521:      else begin   {Keine Datei gef. (Kann n
       ur im Root-Dir passieren)
[586]  522:       for j := 1 to 6 do begin
[H1C]  523:        h1[ya,xa + pred(j) shl 1] := ord(Lee
       r[j]);
[SZ8]  524:        h1[ya,succ(xa + pred(j) shl 1)] := 1
       5
[TU0]  525:       end;
[UX3]  526:       ys := succ(ya);
[TU0]  527:      end;
[TU0]  528:     end;
[000]  529:
[8F6]  530:   procedure dateiwahl;
[EG8]  531:    var  akt, am, m, anz : integer;
[D92]  532:                      ch : char;
[FR4]  533:                    f, rq : boolean;
[DB4]  534:        aq, xq, yq, x1q,
[M34]  535:          y1q, mq, anzq,
[GU4]  536:            xanzq, yanzq : byte;
[000]  537:
[H0C]  538:    procedure MoveLinie( att, a, b, y, c : b
       yte );
[5T4]  539:     var i : integer;
[BH1]  540:      begin
[TL4]  541:       a := pred(a) shl 1;
[T85]  542:       i := (pred(y)*5) shl 5;
[VC2]  543:       b := b shl 1;
[X52]  544:       repeat
[G58]  545:        mem[ screenaddr : i+a ] := c;
[933]  546:        a := succ(a);
[UG9]  547:        mem[ screenaddr : i+a ] := att;
[933]  548:        a := succ(a);
[VC3]  549:       until a >= b;
[TU0]  550:      end;
[000]  551:
[DB5]  552:    procedure ober;
[BH1]  553:      begin
[X94]  554:       attribut := 112;
[Z15]  555:       MoveChr(x,2,'┌');
[C68]  556:       MoveLinie(112,succ(x),xs,y,196);
[1A7]  557:       MoveChr(succ(xs),2,'┐');
[TU0]  558:      end;
[000]  559:
[8N7]  560:    procedure seitenunter;
[W63]  561:     var i : byte;
[BH1]  562:      begin
[X94]  563:       attribut := 112;
[1P8]  564:       for i := 3 to succ(ys) do begin
[UZ4]  565:        MoveChr(x,i,'|');
[LB7]  566:        MoveChr(succ(xs),i,'|');
[TU0]  567:       end;
[H55]  568:       MoveChr(x,ys+2,'└');
[45C]  569:       MoveLinie(attribut,succ(x),xs,ys+2,19
       6);
[D97]  570:       MoveChr(succ(xs),ys+2,'┘');
[TU0]  571:      end;
[000]  572:
[KK5]  573:     begin (dateiwahl)
[B84]  574:      attribut := a;
[YN1]  575:      akt := 1;
[TK1]  576:      y1q := 1;
[SK1]  577:      x1q := 1;
[8B3]  578:      xq := succ(x);
[DB3]  579:      yq := succ(y);
[UG1]  580:      mq := 15;
[WA2]  581:      anzq := 12;
[TK4]  582:      if Passend > 5 then
[1C2]  583:       xanzq := 5
[A11]  584:      else
[RH4]  585:       if Passend = 0 then
[TB2]  586:        xanzq := 1
[A11]  587:       else
[045]  588:        xanzq := Passend;
[MX4]  589:      yanzq := pred(ys);
[9D2]  590:      rq := true;
[X52]  591:      repeat
[D12]  592:       ys1 := ys+2;
[C9B]  593:       move(mem[screenaddr:160+pred(x) shl 1
       ],h,
[L99]  594:            (pred(ys)*5) shl 5+pred(x) shl 1
       );
```

```
[B84]  595:       attribut := a;
[HA1]  596:       ober;
[CL4]  597:       attribut := a1;
[KZ7]  598:       MoveStr(x,pred(y),'Pfad: ');
[62E]  599:       MoveStr(x+5,pred(y),chr(Laufwerk+64)
       + ':' + pfad);
[HK4]  600:       if Passend < 5 then
[RH4]  601:        if Passend = 0 then
[FP1]  602:         anz := 24
[A11]  603:        else
[XW5]  604:         anz := pred(Passend)*30+24
[A11]  605:       else
[8Z1]  606:        anz := 144;
[VU7]  607:       am := (pred(y)*5) shl 5 + x shl 1;
[5N6]  608:       for i := 1 to ys do begin
[SU4]  609:        m := am + (i*5) shl 5;
[56A]  610:        move(h1[i],mem[screenaddr:m],anz);
[TU0]  611:       end;
[8N3]  612:       seitenUnter;
[6NE]  613:       wahl2(aq,xq,yq,x1q,y1q,8,mq,anzq,xanz
       q,yanzq,rq);
[035]  614:       if n then begin
[1H5]  615:        if not BB then begin
[U04]  616:         altDatei := '';
[S64]  617:         altPfad := '\';
[DL0]  618:        end
[A11]  619:        else
[9F2]  620:         n := false;
[KF5]  621:        auswahlB := true;
[7C1]  622:        exit;
[TU0]  623:       end;
[GK5]  624:       if not rq then begin
[955]  625:        altPfad := pfad;
[22E]  626:        altDatei := Datei[search(x1q,y1q,sei
       te)].name;
[KF5]  627:        auswahlB := true;
[7C1]  628:        exit;
[TU0]  629:       end;
[6M2]  630:      until rq;
[TU0]  631:     end;
[000]  632:
[7M5]  633:    begin (auswahl)
[RZ3]  634:     Window(1,1,80,22);
[NM2]  635:     clrscr;
[RZ3]  636:     Window(1,1,80,25);
[E55]  637:     if Passend >= 5 then
[502]  638:      xs := mxs
[A11]  639:     else
[RH4]  640:      if Passend = 0 then
[EF1]  641:       xs := 17
[A11]  642:      else
[W26]  643:       xs := pred(Passend*15);
[GL3]  644:     moveDateien;
[P33]  645:     dateiwahl;
[8EB]  646:     move(h,mem[screenaddr:160+pred(x) shl
       1],
[L99]  647:          (pred(ys)*5) shl 5+pred(x) shl 1)
       ;
[TU0]  648:    end;
[000]  649:
[1YN]  650:  procedure dateiSuchen( ersten : boolean; v
       ar groesse : real; var att : byte;
[H2C]  651:                         var dateiname : tst
       r80; var datum : tstr8;
[CG6]  652:                         var fehler : boolea
       n );
[36A]  653:   Type       ast = array[1..80] of byte;
[E76]  654:   Var  h1, h2, i : integer;
[N62]  655:                as : ast;
[9G2]  656:                 t : byte;
[X33]  657:              regs : tregs;
[V74]  658:              hilf : string[2];
[BH1]  659:    begin
[Y36]  660:     if ersten then begin
[375]  661:      with regs do begin
[AM7]  662:       dateiname := dateiname + #0;
[XF1]  663:       Ah := $1A;
[1Y3]  664:       DS := Seg(as[1]);
[VX3]  665:       DX := Ofs(as[1]);
[W33]  666:       MSDos(regs);
[1G1]  667:       Ah := $4E;
[CE1]  668:       CX := 30;
[8B6]  669:       DS := Seg(dateiname[1]);
[2B6]  670:       DX := Ofs(dateiname[1]);
[WW2]  671:       MSDos(Regs);
[TU0]  672:      end;
[DL0]  673:     end
[YS2]  674:     else begin
[A14]  675:      regs.ah := $4F;
[W33]  676:      MSDos(regs);
[TU0]  677:     end;
[G56]  678:     fehler := regs.al <> 0;
[ER1]  679:     INT24F;
[ED7]  680:     if not fehler then begin
[F23]  681:      att := as[22];
[PZ7]  682:      Move(as[31],Dateiname[1],13);
[E05]  683:      Dateiname[0] := #13;
[KKC]  684:      dateiname[0] := chr(pred(pos(#0,datein
       ame)));
```

DAS GRAFIKWUNDER

CGX FÜR DEN PC!

CGX (**C**olor **G**raphik **E**xtension) ist eine Turbo Pascal-Toolbox der Superlative: Eine Sammlung von 100 leistungsfähigen Prozeduren, mit denen sowohl der Heimprogrammierer als auch der Profi grafisch beeindruckende und benutzerfreundliche Turbo Pascal-Programme schreiben können, ohne auf Spezialkenntnisse oder zeitraubende Assemblerprogrammierung angewiesen zu sein.

Mit CGX können aus ein und dem gleichen Quelltext lauffähige Programme für 3 hochauflösende PC-Grafikmodi erzeugt werden:

- Schneider PC 1512 Farbgrafik (640x200 bei 16 Farben)
- EGA Farbgrafik (z.B. Schneider PC 1640, Atari) (640x200 bei 16 Farben)
- CGA Hires-Grafik (640x200 bei 2 Farben)

Damit ist es ohne weiteres möglich, auf dem Schneider PC 1512 Software für EGA-Rechner zu entwickeln – und umgekehrt!

Die CGX-Prozeduren arbeiten sehr schnell und effektiv, da sie unter Verwendung modernster Grafikalgorithmen zu einem großen Teil in kompakten 8086 Inline-Assemblercode realisiert wurden. Sie sind in mehreren Bibliotheken zusammengefaßt, die als Include-Files problemlos in eigenen Programmen verwendet werden können:

Die **KERNEL – Bibliothek** stellt den CGX-Betriebssystem-Kern dar und enthält u. a. einen neuen Bildschirmtreiber, der verschiedene Schrifttypen und -größen zur Verfügung stellt, weiterhin Unterstreichen, Fettschrift, Exponenten und Indizes sowie die Textausgabe auf Grafikkoordinaten, Transparent- und XOR-Modus und vieles mehr.

Die **GRAPHIK-Bibliothek** enthält zahlreiche Grafikprozeduren, angefangen bei »Plot« und »Draw« bis hin zu Ellipsenbögen und Rechtecken mit runden Ecken. Weiterhin finden Sie hier eine ganze Palette von Füllprozeduren für Muster und Farben sowie Prozeduren zum Verzerren bzw. Biegen von Bildschirmbereichen.

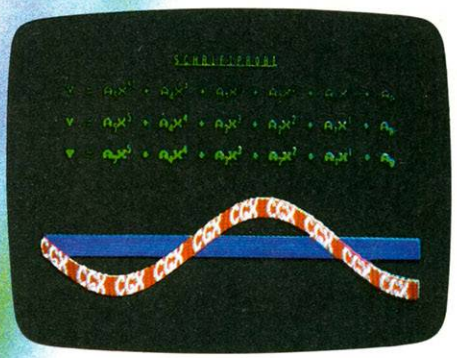
Die **MOUSE-Bibliothek** stellt die Schnittstelle zum Maustreiber dar und erlaubt neben der Abfrage und Kontrolle der Mausposition und -tasten auf komfortable Weise die Gestaltung eigener Mauszeiger. Zusätzlich gibt sie Hilfen zur »Event«-Verarbeitung: »Mausereignisse« können definiert und abgefragt werden.

Die **HARDCOPY-Bibliothek** ermöglicht den Ausdruck beliebiger Bildschirmausschnitte in frei wählbarem Format, wobei die 16 Farben durch verschiedene Muster dargestellt werden.

Die **SCREEN-Bibliothek** enthält Prozeduren, mit denen Bildschirmbereiche im Speicher oder auf Diskette abgelegt und bei Bedarf wieder geladen werden.

Die **WINDOW-Bibliothek** stellt ein System zur Verwaltung von Text- und Grafikfenstern zur Verfügung – die Grundlage für eine moderne Menuetechnik.

Weiterhin umfaßt CGX neben Anwendungsbeispielen und einem ausführlichen Handbuch kommentierte Quellprogramme für einen Füllmuster- und Zeichensatzeditor – eine komfortable Programmierhilfe mit »Pull-Down«-Menues und Mausbedienung.



Neue Gestaltungsmöglichkeiten – auch hier überzeugt CGX

Autor: M. Uphoff

CGX für MS-DOS PC unter Turbo Pascal

Farbenvielfalt durch CGX – mehr als 100 Farben am PC!

199,– DM

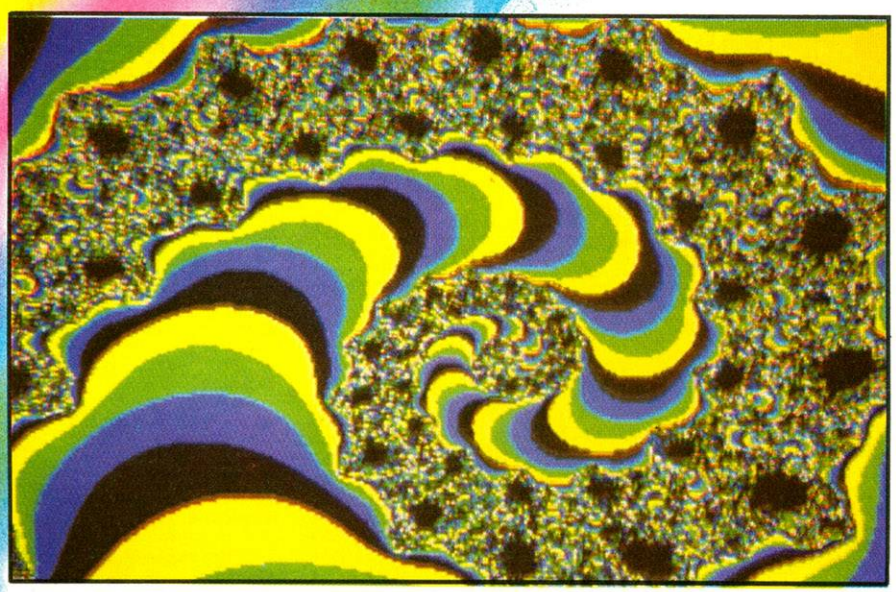
TEXTVERARBEITUNG, TABELLENKALKULATION LOHNBUCHHALTUNG-STATISTIK.....

Überarbeitet? Gönnen Sie sich eine Urlaubsreise in das Land der Fraktale, genießen Sie die phantastische Farb-Fähigkeit Ihres PC.

– eine neue Dimension für die Freunde fraktaler Grafiken: Über 100 (!) Farben stellt Ihnen dieses Programm bei einer Auflösung von 640 × 200 Punkten zur Verfügung. Und das berühmte »Apfelmännchen« braucht dank ausgefeilter Algorithmen nur noch ca. 3 Minuten für seine Entstehung – keine stundenlangen Wartezeiten hindern Sie mehr daran, eine märchenhafte Welt voller Farben und Formen zu entdecken.

Und dazu der Bedienungskomfort:
- Bedienung per Maus und Pulldown-Menues
- Hardcopy auf Knopfdruck
- Speichern von Bildern auf Diskette

FRAKTAL GENERATOR
- Schneider PC 1512 /PC 1640
- Atari PC
- PC's mit EGA-Karte & Microsoftkompatibler Maus.



- Umschalten zwischen verschiedenen Bildern
- Nachträgliches Ändern der Farben
- Vergrößerte Ausschnitte durch Auswahlrahmen
- Rechentiefe bis 9999

FRAKTAL GENERATOR für MS-DOS
- ein **CGX**-Programm von **DMV**

nur **49,– DM**

FRAKTAL GENERATOR
Autor: Matthias Uphoff

– Bitte benutzen Sie die Bestellkarte –

DMV Daten & Medien Verlagsgesellschaft mbH · Postfach 250 · Fuldaer Str. 6 · 3440 Eschwege

```
[2F4]  685:       move(as[27],h2,2);
[OF4]  686:       move(as[29],h1,2);
[CM3]  687:       if h2 < 0 then
[HF3]  688:        h1 := succ(h1);
[V57]  689:       groesse := 6.5536e4 * h1 + h2;
[FS4]  690:       t := as[25] and 31;
[TA4]  691:       str(t:2,datum);
[TY4]  692:       datum := datum+'.';
[WEA]  693:       t := (as[26] and 1) shl 3 + as[25] shr
         5;
[MM3]  694:       str(t:2,hilf);
[9Y6]  695:       datum := datum + Hilf + '.';
[TB5]  696:       t := as[26] shr 1+80;
[MT2]  697:       str(t,Hilf);
[UA6]  698:       datum := datum + Hilf;
[TUO]  699:      end;
[TUO]  700:     end;
[000]  701:
[3RN]  702:  procedure DateiAnfuegen( dat : tstr12; a :
         byte; gr : real; datum : tstr8 );
[5T4]  703:    var i : integer;
[BH1]  704:     begin
[T77]  705:      if boolean(a and 8) then
[7C1]  706:       exit;
[XA6]  707:      if (dat <> '.') then begin
[VK9]  708:        DateiAnzahl := succ(DateiAnzahl);
[TUC]  709:        fillchar(Datei[DateiAnzahl].name[1],1
         2,#32);
[ZT6]  710:        if dat[1] <> '.' then begin
[A54]  711:         i := pos('.',dat);
[SG4]  712:         if i > 0 then begin
[AUD]  713:          move(dat[1],Datei[DateiAnzahl].name
         [1],pred(i));
[62K]  714:          move(dat[i],Datei[DateiAnzahl].name
         [9],byte(dat[0])-pred(i));
[DLO]  715:         end
[A11]  716:         else
[6FF]  717:          move(dat[1],Datei[DateiAnzahl].name
         [1],byte(dat[0]));
[5C9]  718:         Datei[DateiAnzahl].name[0] := #12;
[DLO]  719:        end
[A11]  720:        else
[7WC]  721:         Datei[DateiAnzahl].name := '<Übergeo
         rd.>';
[T67]  722:        Datei[DateiAnzahl].a := a;
[L18]  723:        Datei[DateiAnzahl].gr := gr;
[BKA]  724:        Datei[DateiAnzahl].datum := datum;
[HY7]  725:        Passendgr := Passendgr + gr;
[TG9]  726:        Datei[DateiAnzahl].passend := 1;
[TUO]  727:      end;
[TUO]  728:     end;
[000]  729:
[KGE]  730:  procedure suchen( dat : tstr12; var f : bo
         olean );
[4U3]  731:    var     a : byte;
[4C2]  732:          gr : real;
[5Y3]  733:        datum : tstr8;
[BH1]  734:     begin
[K6B]  735:      dateisuchen(true,gr,a,dat,datum,f);
[V75]  736:      if not f then begin
[X52]  737:       repeat
[KV8]  738:        DateiAnfuegen(dat,a,gr,datum);
[SZA]  739:        dateisuchen(false,gr,a,dat,datum,f);
[RF2]  740:       until f;
[TUO]  741:      end;
[AD2]  742:      Seite := 1;
[0U7]  743:      Passend := DateiAnzahl;
[DX9]  744:      MaxSeite := succ(Passend div 90);
[EZ3]  745:      if BSort then
[TN2]  746:       sorter;
[TUO]  747:     end;
[000]  748:
[H95]  749:  procedure bild;
[6G6]  750:   var str4 : string[4];
[BH1]  751:    begin
[NM2]  752:     clrscr;
[V04]  753:     attribut := 1;
[XMA]  754:     MoveStr(50,25,'(c) A. Baqué   Bonn, Nov
         .'87');
[324]  755:     attribut := 9;
[P98]  756:     MoveStr(32,25,'SDIR Version 1.4');
[MC4]  757:     attribut := 15;
[HA5]  758:     MoveStr(1,25,'<ESC>');
[D94]  759:     MoveChr(44,24,'N');
[ZK4]  760:     MoveStr(1,24,'F1');
[594]  761:     MoveChr(21,24,'S');
[994]  762:     MoveChr(61,24,'F');
[NP7]  763:     MoveStr(70,24,#24#25#27#26#32#17#217);
[CC4]  764:     attribut := 32;
[U0A]  765:     MoveStr(7,25,'Programm verlassen');
[U16]  766:     MoveStr(62,24,'ilter');
[3K8]  767:     MoveStr(45,24,'eues Laufwerk');
[LG8]  768:     MoveStr(4,24,'Speicher-Stat.');
[D7A]  769:     MoveStr(22,24,'ortieren nach Namen');
[836]  770:     MoveStr(1,23,'Größe :');
[PF5]  771:     MoveStr(18,23,'Byte');
[Z09]  772:     MoveStr(25,23,'Dateiattribut :');
[Y16]  773:     MoveStr(46,23,'Datum :');
```

```
[TUO]  774:     end;
[000]  775:
[A9D]  776:  procedure neu( dat : tstr12; var weg : boo
         lean );
[BH1]  777:    begin
[LX2]  778:     weg := false;
[035]  779:     if n then begin
[9F2]  780:      n := false;
[X56]  781:      aktdrive(Laufwerk);
[TUO]  782:     end;
[7D5]  783:     auswahlB := false;
[HU7]  784:     if dat = '<Übergeord.>' then
[HV1]  785:      dat := '..'
[A11]  786:     else
[SP5]  787:      while pos(' ',dat) <> 0 do
[M57]  788:       delete(dat,pos(' ',dat),1);
[VG4]  789:     if pfad <> '\' then
[WA5]  790:      pfad := pfad + '\' + dat
[A11]  791:     else
[104]  792:      pfad := pfad + dat;
[9PD]  793:     if (pfad[length(pfad)] = '\') and (pfad
         <> '\') then
[H58]  794:      delete(pfad,length(pfad),1);
[NM2]  795:     clrscr;
[826]  796:     chdirectory(pfad,f);
[025]  797:     if f then begin
[W52]  798:      pfad := '';
[XE2]  799:      weg := true;
[TUO]  800:     end;
[F91]  801:     bild;
[CB6]  802:     if dat = '..' then begin
[EB4]  803:      aktdir( pfad, 0 );
[D55]  804:      insert('\',pfad,1)
[TUO]  805:     end;
[1M4]  806:     if pfad = '' then
[PE2]  807:      pfad := '\';
[B94]  808:     DateiAnzahl := 0;
[DR4]  809:     if not(weg) then
[AY3]  810:      suchen('*.*',f);
[XF8]  811:     if BF and not(weg) then begin
[706]  812:      einzeln(Referenz);
[TUO]  813:     end;
[TUO]  814:    end;
[000]  815:
[BH1]  816:   begin
[1H5]  817:    if not BB then begin
[H84]  818:     Passendgr := 0.0;
[VL3]  819:     Passend := 0;
[BR4]  820:     pfad := altpfad;
[SS5]  821:     neu(altdatei,quit);
[DLO]  822:    end
[YS2]  823:    else begin
[PH2]  824:     BB := false;
[7D5]  825:     auswahlB := false;
[0X3]  826:     quit := false;
[TUO]  827:    end;
[2D7]  828:    if not(quit) then begin
[892]  829:     if BF then
[706]  830:      einzeln(Referenz);
[9Y2]  831:     auswahl;
[DLO]  832:    end
[A11]  833:    else
[7D5]  834:     auswahlB := false;
[TUO]  835:   end;
[000]  836:
[AFC]  837: begin   {CBreak:=false bringt hier nichts}
[FTA]  838:  ctrlbreak(true,ctrlbreakstatus);
[MP7]  839:  ctrlbreak(false,trueB);
[0B3]  840:  wx := wherex;
[6B3]  841:  wy := wherey;
[7C5]  842:  CursorFrage(0,Ca,Ce);
[DK2]  843:  screen;
[SK3]  844:  Cursor(32,32);
[8S8]  845:  move(mem[screenaddr:0],h,4000);
[WZ1]  846:  int24S;
[SA4]  847:  altpfad := '\';
[U44]  848:  altdatei := '';
[GM4]  849:  BSort := false;
[PH2]  850:  BB := false;
[YH2]  851:  BF := false;
[9F2]  852:  n := false;
[T76]  853:  Referenz := '????????.???';
[LV7]  854:  aktdrive(saveLaufwerk);
[D68]  855:  Laufwerk := SaveLaufwerk;
[586]  856:  aktdir( savepfad, 0 );
[X52]  857:  repeat
[8A1]  858:   wahl;
[BT5]  859:  until not auswahlB;
[PZ1]  860:  int24R;
[B74]  861:  gotoxy(wx,wy);
[3W3]  862:  Cursor(Ca,Ce);
[OL8]  863:  move(h,mem[screenaddr:0],4000);
[XU7]  864:  Newdrive(saveLaufwerk);
[K09]  865:  chdirectory( '\'+savepfad, f );
[99B]  866:  CtrlBreak(false,CtrlBreakStatus);
[5T0]  867: end.
```

Listing 15. »Sdir« – eine sehr komfortable Directory-Routine in Turbo Pascal

```
[375]    0: {$I-,K-,D-,R-,V-,C-,U-}
[000]    1:
[DUG]    2: {MDIR.PAS 20/21.11.87, (Geändert für DOS in
           ternational 8.1.88)}
[000]    3:
[WP3]    4: {$I LIBRARY.INC}
[R64]    5: {$I WINDOW.INC}
[XU3]    6: {$I LIBRARYE.INC}
[000]    7:
[SZ8]    8: type TSetOfChar = Set of Char;
[000]    9:
[WP3]   10: {$I READSTR.INC}
[4P3]   11: {$I TOOLS.INC}
[000]   12:
[2P4]   13: procedure dir;
[CKK]   14:  const      str43 = 'Taste drücken (<'#24'>
           Zurück  <ESC> Abbrechen)';
[EL9]   15:             str28 = 'NDIR  (c) 1987 A.Baqué
           , Bonn';
[V3N]   16:    ERKL1 = '<ESC> Abbruch  Edit.: <'#27','#2
           6'> Bewegen, <Einfg/Ins> Insert/Overwrite,';
[SML]   17:    ERKL2 = '<Entf/Del> Unterm Cursor weg, <R
           uck./Back.> Vorm Cursor weg,';
[3LM]   18:    ERKL3 = '<F1> Alles weg, <F2> Ab Cursor l
           öschen, <F3> Vor Cusor löschen';
[CML]   19:    ERKL4 = 'Parameter (ohne Leerschritt) : /
           1../8 (Darstellung)./S (Sort.)';
[E17]   20:          optionen : integer = 7;
[AD1]   21:                a7 = 7;
[BF1]   22:               ys1 = 8;
[KK1]   23:              a112 = 112;
[GO3]   24:            MaxDat = 512;
[RD8]   25:  type      tstr80 = string[ 80 ];
[WH7]   26:       tsetofchar = set of char;
[YP2]   27:                TT = record
[1C4]   28:                                    name : st
           ring[12];
[AH4]   29:                        jahr, monat, tag,
[B15]   30:                          stunde, minute,
[888]   31:                        sekunde, attribut : by
           te;
[4C2]   32:                                      gr : re
           al;
[TUO]   33:                      end;
[35A]   34:         Schneller = record          {43 Byt
           es}
[WO5]   35:                        case boolean of
[FX7]   36:                         true : (  v : array[1
           ..26] of byte;
[7AO]   37:                                  a1,
[2W3]   38:                                   a2 : integer
           );
[OB8]   39:                        false : ( a0 : array[1
           ..29] of byte;
[CE4]   40:                                   a7 : string[
           13] );
[TUO]   41:                      end;
[NRA]   42:  var         anzahl, anzxss, i, c : integer
           ;
[OW7]   43:        speicherFrei, speicherKap,
[9LA]   44:           speicherBelegt, groesse : real;
[4S4]   45:                                   as : schnell
           er;
[B54]   46:                                 str9 : string[
           9];
[Y24]   47:                                 str2 : string[
           2];
[5Z3]   48:                               str255 : tstr255
           ;
[PY2]   49:                                 c5, t : byte;
[7G3]   50:                BSort, b1, b2, b3,
[XV7]   51:                exist, error, raus : boolean
           ;
[AT3]   52:                                 str80 : tstr80;
[X33]   53:                                  regs : tregs;
[L18]   54:                                  save : array[1
           ..MaxDat] of TT;
[C58]   55:                                    hh : array[1
           ..7,1..140] of byte;
[6PE]   56:  const                          filter : tsetofc
           har = [ ' ', '!', '#'..'*', ','..
[164]   57:
                 ':', '?'..'\', '^'..'{',
[332]   58:
                 '}'..'~'];
[000]   59:
[DT5]   60:  procedure unter;
[W63]   61:   var i : byte;
[BH1]   62:    begin
[GM4]   63:     attribut := a7;
[GL6]   64:     SchreibeZeichen('└');
[1L4]   65:     attribut := a112;
[P16]   66:     SchreibeSatz(str43);
[GM4]   67:     attribut := a7;
[PA6]   68:     for i := 44 to anzxss+2 do
[SL6]   69:      SchreibeZeichen('-');
[2N6]   70:     SchreibeZeichen('┘');
[TUO]   71:    end;
```

```
[000]   72:
[654]   73:  procedure st;
[257]   74:   var str10 : string[10];
[BH1]   75:    begin
[GD6]   76:     fillchar(str10[1],10,#32);
[763]   77:     str10[0] := #10;
[DED]   78:     MoveStr(3,5,'Gesamter Speicher :'+str10
           +'Byte');
[X7C]   79:     MoveStr(3,6,'Verzeichnisgroße  :'+str10
           +
[CMA]   80:                 'Byte  Freier Speicher :'+s
           tr10+'Byte');
[C93]   81:     str10[0] := #5;
[LT8]   82:     MoveStr(37,5,str10+'Datei(en)');
[1H5]   83:     MoveStr(53,5,'Name:');
[TUO]   84:    end;
[000]   85:
[DXN]   86:  procedure hilfdir( var exist : boolean; va
           r n : boolean; var i : integer );
[LR6]   87:   var c1, c2, c3, c4 : integer;
[B54]   88:                 str9 : string[9];
[X33]   89:                 regs : tregs;
[D92]   90:                   ch : char;
[ER3]   91:                  bah : boolean;
[BH1]   92:    begin
[XY1]   93:     unter;
[FR4]   94:     exist := false;
[1L4]   95:     attribut := a112;
[035]   96:     if n then begin
[9F2]   97:      n := false;
[6X6]   98:      speicher(c1, c2, c3, c4, c5);
[HC7]   99:      speicherFrei := 1.0 * c1 * c3;
[ZXA]  100:      speicherKap := 1.0 * c4 * SpeicherFrei
           ;
[N89]  101:      SpeicherFrei := SpeicherFrei * c2;
[TUO]  102:     end;
[SR1]  103:     Adr(22,5);
[TD7]  104:     MoveReal(9,0,SpeicherKap);
[VS1]  105:     Adr(37,5);
[MP5]  106:     MoveInt(4,anzahl);
[SR1]  107:     Adr(22,6);
[UA8]  108:     MoveReal(9,0,SpeicherBelegt);
[TS1]  109:     Adr(55,6);
[ZP7]  110:     MoveReal(9,0,SpeicherFrei);
[X52]  111:     repeat
[TW2]  112:      bah := false;
[D92]  113:      read2(ch);
[UY5]  114:      if ch = #0 then begin
[D92]  115:       read2(ch);
[RAC]  116:       if (ch = #72) and (succ(c shl 4) < i)
           then begin
[N94]  117:        i := i - (c shl 5);
[VO3]  118:        if i < 1 then
[ED1]  119:         i := 1;
[DLO]  120:       end
[A11]  121:       else
[NY8]  122:        bah := (succ(c shl 4) >= i) and (ch=
           #72);
[DLO]  123:      end
[A11]  124:      else
[R54]  125:       if ch = #27 then
[344]  126:        exist := true;
[154]  127:      if (ch<>#72) then
[BO6]  128:       bah := (i=succ(anzahl));
[307]  129:     until exist or not bah;
[TUO]  130:    end;
[000]  131:
[DX5]  132:  procedure nofiles;
[PT4]  133:   var i, l : integer;
[663]  134:         n : boolean;
[BH1]  135:    begin
[1D2]  136:     n := true;
[AX2]  137:     l := memadr;
[GM4]  138:     attribut := a7;
[2DD]  139:     SchreibeSatz('| Keine Datei(en) gefunde
           n !');
[OB6]  140:     for i := 31 to anzxss+5 do
[LB6]  141:      SchreibeZeichen(#32);
[PK6]  142:     SchreibeZeichen('|');
[SV3]  143:     memadr := l+160;
[MX5]  144:     hilfdir(exist,n,i);
[TUO]  145:    end;
[000]  146:
[879]  147:  procedure sort( v, b : integer );
[UC5]  148:   var i, j, m : integer;
[WC4]  149:              vg : string[12];
[FS1]  150:               h : TT;
[BH1]  151:    begin
[UG7]  152:     vg := save[(v+b) shr 1].name;
[CU1]  153:     i := v;
[NP1]  154:     j := b;
[X52]  155:     repeat
[AF7]  156:      while save[i].name < vg do
[943]  157:       i := succ(i);
[PF7]  158:      while save[j].name > vg do
[733]  159:       j := pred(j);
[515]  160:      if i <= j then begin
[M93]  161:       h := save[i];
```

```
[MS5]  162:        save[i] := save[j];
[MP3]  163:        save[j] := h;
[943]  164:        i := succ(i);
[733]  165:        j := pred(j);
[TUO]  166:       end;
[M53]  167:      until i > j;
[B13]  168:      if j > v then
[PE2]  169:       sort(v,j);
[V03]  170:      if i < b then
[8C2]  171:       sort(i,b);
[TUO]  172:     end;
[000]  173:
[KM5]  174:  procedure zeige;
[4OH]  175:   const     q : array[1..5] of String[3] =(
         'R/O','Hid','Sys','Dir','Arc');
[L13]  176:   var max, i,
[E35]  177:       hilf, l : integer;
[TB4]  178:            s9 : string[9];
[B55]  179:      exist, n : boolean;
[TZ3]  180:         a, anz : byte;
[000]  181:
[Y97]  182:   procedure op8( e : byte );
[055]  183:    var s : string[20];
[BH1]  184:     begin
[EK5]  185:      fillchar(s[1],20,#32);
[R62]  186:      s[0] := #20;
[2FD]  187:      if boolean(e and  1) then  move(q[1][1
         ],s[02],3);
[SFD]  188:      if boolean(e and  2) then  move(q[2][1
         ],s[06],3);
[CFD]  189:      if boolean(e and  4) then  move(q[3][1
         ],s[10],3);
[7HD]  190:      if boolean(e and 16) then  move(q[4][1
         ],s[14],3);
[KHD]  191:      if boolean(e and 32) then  move(q[5][1
         ],s[18],3);
[GM4]  192:      attribut := a7;
[815]  193:      SchreibeSatz(s);
[TUO]  194:     end;
[000]  195:
[637]  196:   procedure MM( t : byte );
[BH1]  197:     begin
[RF3]  198:      str(t:2,str2);
[6K3]  199:      if t < 10 then
[2M3]  200:       str2[1] := #48;
[AV5]  201:      SchreibeSatz(str2);
[TUO]  202:     end;
[000]  203:
[BH1]  204:    begin
[EZ3]  205:     if BSort then
[D34]  206:      sort(1,anzahl);
[HT3]  207:     memadr := 1126;
[GM4]  208:     attribut := a7;
[4N6]  209:     SchreibeZeichen('┌');
[1L4]  210:     attribut := a112;
[SC6]  211:     SchreibeSatz('Seite  1');
[GM4]  212:     attribut := a7;
[R86]  213:     for i := 9 to anzxss + 2 do
[SL6]  214:      SchreibeZeichen('-');
[EL6]  215:     SchreibeZeichen('┐');
[LU3]  216:     memadr := 1286;
[XH4]  217:     if anzahl > 16 then
[9Y1]  218:      max := 3846
[A11]  219:     else
[6R7]  220:      max := memadr+(anzahl*5) shl 5;
[1D2]  221:     n := true;
[PP1]  222:     anz := 0;
[ED1]  223:     i := 1;
[2P4]  224:     if anzahl = 0 then
[1Y2]  225:      nofiles
[A11]  226:     else
[X52]  227:     repeat
[Y96]  228:      if anz = 0 then begin
[AX2]  229:       l := memadr;
[GM4]  230:       attribut := a7;
[GU5]  231:       SchreibeSatz('| ');
[DLO]  232:      end
[A11]  233:      else
[BU4]  234:       SchreibeSatz('  ');
[U3D]  235:      if boolean(save[i].attribut and 16) th
         en
[KN1]  236:       a := a112
[A11]  237:      else
[72C]  238:      if boolean(save[i].attribut and 4) or
[W9B]  239:         boolean(save[i].attribut and 2) the
         n
[AN1]  240:       a := a7
[A11]  241:      else
[WB1]  242:       a := 15;
[B84]  243:      attribut := a;
[388]  244:      SchreibeSatz(save[i].name);
[Z92]  245:      if b1 then
[M14]  246:       if a = 112 then
[706]  247:        SchreibeSatz('  <DIR>  ')
[A11]  248:       else
[6A7]  249:        MoveReal(9,0,save[i].gr);
[EL4]  250:      if b2 then begin
[2N7]  251:       MoveByte(3,save[i].tag);
[2T4]  252:       SchreibeSatz('.');
[XF5]  253:       MM(save[i].monat);
[SA7]  254:       str(save[i].jahr+1980:4,s9);
[AG5]  255:       SchreibeSatz('.'+s9);
[TUO]  256:      end;
[LL4]  257:      if b3 then begin
[WM8]  258:       MoveByte(3,save[i].stunde);
[PV4]  259:       SchreibeSatz(':');
[MT5]  260:       MM(save[i].minute);
[PV4]  261:       SchreibeSatz(':');
[L26]  262:       MM(save[i].sekunde);
[TUO]  263:      end;
[RD5]  264:      if optionen = 7 then
[E29]  265:       MoveByte(3,save[i].attribut)
[A11]  266:      else
[YD5]  267:      if optionen = 8 then
[6X6]  268:       op8(save[i].attribut);
[943]  269:      i := succ(i);
[9D4]  270:      anz := succ(anz);
[1B7]  271:      if (anz = c) then begin
[GM4]  272:       attribut := a7;
[BU4]  273:       SchreibeSatz(' |');
[PP1]  274:       anz := 0;
[SV3]  275:       memadr := 1+160;
[TUO]  276:      end;
[CLD]  277:      if (memadr = max) or (i=succ(anzahl))
         then begin
[TMA]  278:       if (anz < c) and (anz > 0) then begin
[8Y6]  279:        hilf := 1+((anzxss+3) shl 1);
[GM4]  280:        attribut := a7;
[UY6]  281:        while memadr <> hilf do
[LB6]  282:         SchreibeZeichen(#32);
[PK6]  283:        SchreibeZeichen('|');
[SV3]  284:        memadr := 1+160;
[TUO]  285:       end;
[PP1]  286:       anz := 0;
[MX5]  287:       hilfdir(exist,n,i);
[GRA]  288:       move(h[9,1],mem[screenaddr:1280],2720
         );
[RS3]  289:       memadr := 1140;
[1L4]  290:       attribut := a112;
[PW6]  291:       MoveInt(2,succ(i shr 4));
[LU3]  292:       memadr := 1286;
[TUO]  293:      end;
[7C3]  294:     until exist;
[TUO]  295:    end;
[000]  296:
[BH1]  297:   begin
[LZ7]  298:    if ParamCount = 0 then begin
[1B9]  299:     MakeWindow( 3, a112, 15, 32, 8, 1, 70,
         7, hh );
[1L4]  300:     attribut := a112;
[AT2]  301:     SetAdr(9,1);
[ZOC]  302:     SchreibeSatz( str28 ); NextLine; NextLi
         ne;
[4U8]  303:     SchreibeSatz( ERKL1 ); NextLine;
[7U8]  304:     SchreibeSatz( ERKL2 ); NextLine;
[AU8]  305:     SchreibeSatz( ERKL3 ); NextLine;
[8T5]  306:     SchreibeSatz( ERKL4 );
[NA2]  307:     str255 := '';
[MC4]  308:     attribut := 15;
[FRB]  309:     readStr( str255, filter, 67, 9, 2, true
         );
[1L4]  310:     attribut := a112;
[A76]  311:     if (str255[0] = #255) then
[7C1]  312:      exit;
[DLO]  313:    end
[YS2]  314:    else begin
[FE5]  315:     str255 := ParamStr(1);
[4W9]  316:     for anzahl := 1 to length(str255) do
[4ZB]  317:      str255[anzahl] := upcase(str255[anzahl
         ]);
[TUO]  318:    end;
[H04]  319:    optionen := 7;
[GM4]  320:    BSort := false;
[1L4]  321:    attribut := a112;
[ZT3]  322:    cursor( 32, 32 );
[EFF]  323:    if ( str255[ 0 ] = #0 ) or ( str255[ byt
         e( str255[ 0 ] ) ] = '\' ) or
[C6A]  324:       ( str255[ byte( str255[ 0 ] ) ] = ':'
         ) then
[SY4]  325:     str255 := str255 + '*.*';
[BW5]  326:    if str255[0] > #0 then
[22G]  327:     while (str255[0] > #0) and (str255[pred
         (byte(str255[0]))]='/') and
[U4C]  328:           (str255[byte(str255[0])] in ['1'.
         .'8','S']) do begin
[4EB]  329:      if str255[byte(str255[0])] in ['1'..'8
         '] then
[7OD]  330:       optionen := ord( str255[ byte( str255
         [ 0 ] ) ] ) - 48
[A11]  331:      else
[414]  332:       BSort := true;
[TP9]  333:      str255[ 0 ] := chr( ord( str255[ 0 ] )
         - 2 );
[NYB]  334:      if ( str255[ byte( str255[ 0 ] ) ] = '
         :' ) then
[475]  335:       str255 := str255 + '\';
[GTG]  336:      if ( str255[ 0 ] = #0 ) or ( str255[ b
         yte( str255[ 0 ] ) ] = '\' ) then
```

```
[SY4]  337:        str255 := str255 + '*.*';
[TU0]  338:      end;
[G35]  339:      case optionen of
[BU8]  340:          1 : begin c := 5; anzxss := 68; end;
[6U8]  341:          2 : begin c := 3; anzxss := 67; end;
[MT8]  342:          3 : begin c := 3; anzxss := 73; end;
[XY8]  343:       4, 5 : begin c := 2; anzxss := 66; end;
[WS8]  344:          6 : begin c := 1; anzxss := 41; end;
[FT8]  345:          7 : begin c := 1; anzxss := 44; end;
[8T8]  346:          8 : begin c := 1; anzxss := 61; end;
[TU0]  347:      end;
[7U9]  348:      MakeWindow( 4, a112, a112, 32, 2, 3, 69,
           5, hh );
[1L4]  349:      attribut := a112;
[CG0]  350:      st;
[375]  351:      with regs do begin
[NG1]  352:       ah := $19;
[W93]  353:       msdos(regs);
[6P2]  354:       c5 := al + 65;
[622]  355:       str2 := '';
[KD6]  356:       if str255[2] = ':' then
[AA7]  357:        c5 := ord(upcase(str255[1]));
[PD2]  358:       i := c5 - 64;
[353]  359:       Sicherung;
[594]  360:       aktdir(str80,i);
[934]  361:       EntSicherung;
[DP1]  362:       adr(3,4);
[ER1]  363:       INT24F;
[584]  364:       if BREAK24H then
[7C1]  365:        exit;
[YM4]  366:       str9 := chr(c5) + ':';
[5B6]  367:       if str255[2] <> ':' then
[HV5]  368:        SchreibeSatz(str9);
[SHE]  369:       if (str255[1] <> '\') and (str255[3] <>
           '\') then begin
[8D6]  370:         SchreibeZeichen('\');
[3F6]  371:         if (str80[0] <> #0) then
[ZX6]  372:          SchreibeSatz(str80+'\');
[TU0]  373:       end;
[V96]  374:       SchreibeSatz(str255);
[RLA]  375:       xygoto;                {Position sichern}
[LP4]  376:       str255 := str255 + #0;
[TS4]  377:       str9 := str9 + '\*.*' + #0;
[XF1]  378:       Ah := $1A;
[NC4]  379:       DS := Seg(as.v[1]);
[FC4]  380:       DX := Ofs(as.v[1]);
[W33]  381:       MSDos(regs);
[1G1]  382:       Ah := $4E;
[MD1]  383:       CX := 8;
[ND4]  384:       DS := Seg(str9[1]);
[FD4]  385:       DX := Ofs(str9[1]);
[WW2]  386:       MSDos(Regs);
[Z45]  387:       if al = 0 then begin
[Z63]  388:        as.a7[0] := #13;
[WF9]  389:        as.a7[0] := chr(pred(pos(#0,as.a7)));
[BU1]  390:        adr(58,5);
[SZ5]  391:        SchreibeSatz(as.a7);
[DL0]  392:       end
[A11]  393:       else
[FA7]  394:        MoveStr(58,5,'Kein Name !');
[1G1]  395:       Ah := $4E;
[5E1]  396:       CX := 23;
[OP4]  397:       DS := Seg(str255[1]);
[UN4]  398:       DX := Ofs(str255[1]);
[WW2]  399:       MSDos(Regs);
[ER1]  400:       INT24F;
[5U4]  401:       if Break24H then
[7C1]  402:        exit;
[839]  403:       if (al = 0) and ( as.v[22] = 16 ) and
[2MD]  404:          ( str255[ pred( byte( str255[ 0 ] )
           ) ] <> '*' ) then begin
[A79]  405:        delete( str255, length( str255 ), 1 );
[ACB]  406:        insert( '\*.*', str255, succ( length(
           str255 ) ) );
[LP4]  407:        str255 := str255 + #0;
[UU5]  408:        adr(wherex,wherey);
[RF5]  409:        SchreibeSatz( '\*.*');
[1G1]  410:        Ah := $4E;
[5E1]  411:        CX := 23;
[OP4]  412:        DS := Seg(str255[1]);
[UN4]  413:        DX := Ofs(str255[1]);
[WW2]  414:        MSDos(Regs);
[ER1]  415:        INT24F;
[5U4]  416:        if Break24H then
[7C1]  417:         exit;
[TU0]  418:       end;
[TU0]  419:      end;
[SW2]  420:                anzahl := 0;
[Y56]  421:       SpeicherBelegt := 0.0;
[V7K]  422:                  b1 := (optionen in [2,5..8]
           ); {Zugriff über Booleans schneller}
[K56]  423:                  b2 := (optionen in [3..8]);
[Z27]  424:                  b3 := (optionen in [4,6,7,8
           ]);
[R17]  425:      while regs.al = 0 do begin
[PV6]  426:       anzahl := succ(anzahl);
[XUB]  427:       if boolean(as.v[22] and 16) then begin
[DL0]  428:       end
[81N]  429:       else begin  (Vor Namenserkennung, da hi
           er Byte 30 zersort wird !!!)
[9B4]  430:         if as.a1 < 0 then
[9S6]  431:          as.a2 := succ(as.a2);    {+65536}
[B38]  432:         groesse := 6.5536e4 * as.a2 + as.a1;
[Z92]  433:         if b1 then
[6E8]  434:          save[anzahl].gr := groesse;
[6LD]  435:         speicherBelegt := speicherBelegt + gro
           esse;
[TU0]  436:       end;
[Z63]  437:       as.a7[0] := #13;
[O16]  438:       t := pred(pos(#0,as.a7));
[TM4]  439:       as.a7[0] := chr(t);
[6UA]  440:       fillchar(save[anzahl].name[1],12,#32);
[DN4]  441:       i := pos('.',as.a7);
[SG4]  442:       if i > 1 then begin
[RED]  443:        move(as.a7[1],save[anzahl].name[1],pre
           d(i));
[U5E]  444:        move(as.a7[i],save[anzahl].name[9],t-p
           red(i));
[DL0]  445:       end
[A11]  446:       else
[113]  447:       if i = 1 then
[VK5]  448:        if as.a7[2] = '.' then
[7X9]  449:         save[anzahl].name := '<Übergeord.>'
[A11]  450:        else
[RV9]  451:         save[anzahl].name := '<Aktuelles >'
[A11]  452;       else
[ONB]  453:        move(as.a7[1],save[anzahl].name[1],t);
[HG7]  454:       save[anzahl].name[0] := #12;
[EL4]  455:       if b2 then begin
[4BA]  456:        save[anzahl].tag := (as.v[25] and 31);
[1WG]  457:        save[anzahl].monat := (as.v[26] and 1)
           shl 3 + as.v[25] shr 5; {and 224}
[S1E]  458:        save[anzahl].jahr := as.v[26] shr 1;
                     {Jahr}
[TU0]  459:       end;
[LL4]  460:       if b3 then begin
[REF]  461:        save[anzahl].stunde := (as.v[24] shr 3
           );              {Stunde}
[7UK]  462:        save[anzahl].minute := (as.v[24] and 7
           ) shl 3 + (as.v[23] shr 5); {shr 5}
[59D]  463:        save[anzahl].sekunde := (as.v[23] and
           31) shl 1;
[TU0]  464:       end;
[NOK]  465:       save[anzahl].attribut := as.v[22];  {Wi
           rd immer benötigt}
[A14]  466:       regs.ah := $4F;
[W33]  467:       MSDos(regs);
[ER1]  468:       INT24F;
[5U4]  469:       if Break24H then
[7C1]  470:        exit;
[TU0]  471:      end;
[PV1]  472:      zeige;
[TU0]  473:    end;
[000]  474:
[AFC]  475:   begin   {CBreak:=false bringt hier nichts}
[FTA]  476:    ctrlbreak(true,ctrlbreakstatus);
[MP7]  477:    ctrlbreak(false,trueB);
[0B3]  478:    wx := wherex;
[6B3]  479:    wy := wherey;
[7C5]  480:    CursorFrage(0,Ca,Ce);
[DK2]  481:    screen;
[G93]  482:    Cursor(1,13);
[ZV9]  483:    move( mem[ screenaddr : 0 ], h, 4000 );
[WZ1]  484:    int24S;
[SV0]  485:    dir;
[PZ1]  486:    int24R;
[L0A]  487:    move( h, mem[ screenaddr : 0 ], 4000 );
[B74]  488:    gotoxy(wx,wy);
[3W3]  489:    Cursor(Ca,Ce);
[99B]  490:    CtrlBreak(false,CtrlBreakStatus);
[5T0]  491:   end.
```

Listing 16. Der Quellcode für die Directory-Routine »Mdir«

Tastaturbelegung selbst programmiert

Die Programmierung der Tasten ist eine langwierige Angelegenheit, wenn Sie dies über den Prompt-Befehl realisieren wollen. Es wäre eine feine Sache, müßte man die ständig vorkommenden Zeichenfolgen nicht immer wieder eingeben, sondern könnte diese mit einem einfachen Tastendruck erscheinen lassen. Genau diese Programmierung der Tasten erledigt das kleine Programm »Tasten« (Listing 17), das die editierbare Eingabe von Zeichen erlaubt. Später wandelt ein weiteres Programm die Tastaturbelegung in eine Batch-Datei um (»TastBat«, Listing 18). Den gesamten Vorgang erleichtert der Einsatz der Window-Technik enorm. Ein drittes Programm ermöglicht die schnelle Übersicht über alle Tasten, die programmiert

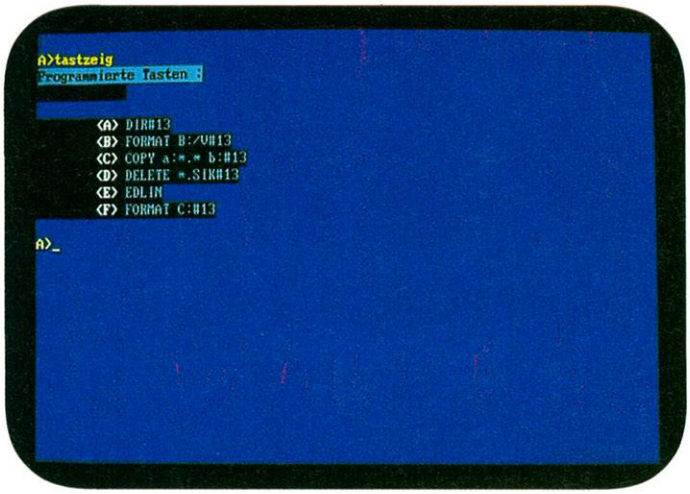


Bild 8. Das Turbo Pascal-Programm »TastZeig« zeigt Ihnen die Tastaturbelegung

wurden, ohne daß der Tastaturcode geläufig sein muß (»TastZeig«, Listing 19). Die Programme übersetzen dem Benutzer die Tastaturcodes in verständliche Zeichen (Bild 8). Die Eingabe der zu programmierenden Tasten ist wirklich kinderleicht, so daß sich eine eingehende Erklärung des Programms erübrigt (Bild 9 und 10). Wundern Sie sich übrigens nicht, wenn sich nach dem Verlassen des Programms »Tasten« auf Ihrem Datenträger eine Datei mit dem Namen »tast76sp.trg« befindet, denn darin werden alle Tastaturbelegungen gespeichert. Bei einem erneuten Aufruf des Programms sehen Sie Ihre bisher programmierten Tasten und können diese gegebenenfalls erweitern. Ohne diese Datei sind die Programme »TastBat« und »TastZeig« nicht lauffähig. Für die Installation der »Batch«-Datei durch »TastBat« ist es notwendig, daß sich in der Konfigurationsdatei »config.sys« die Anweisung »device = ansi.sys« befindet (»ansi«-Treiber installiert). Ohne diese Anweisung lassen sich keine Tasten belegen.

```
[L6K]    0: {tasten.pas 16/17.8.87 (Geändert für DOS in
           ternational 8.1.88)}
[000]    1:
[G65]    2: {$I-,R-,D-,V-,K-,C-,U-}
[000]    3:
[WP3]    4: {$I LIBRARY.INC}
[R64]    5: {$I WINDOW.INC}
[XU3]    6: {$I LIBRARYE.INC}
[000]    7:
[E67]    8: type                    tr = record
[FG4]    9:                               text : string[6
           5];
[TG3]   10:                              taste : byte;
[TU0]   11:                            end;
[GY7]   12:                   tTexte = array[1..76] of t
           r;
[WH7]   13:               tsetofchar = set of char;
[4G7]   14: var                 regs : tregs;
[PB6]   15:               TextAnzahl : integer;
[6P7]   16:                       h1 : array[1..25,1..16
           0] of byte;
[FS4]   17:                    Texte : tTexte;
[000]   18:
[RHP]   19: procedure reading( bereich : tsetofchar; an
           zahl, yanz, x1, y1 : integer;
[EKB]   20:                    var art : byte; Nurgross
           : boolean );
[RD3]   21: { !!!!!!!!!! HINWEIS:
[F4E]   22:   zum Abtippen : Basiert auf READSTR (READST
           R.INC) !!!!!!!!!! }
[GLA]   23:   const                      t : array[1..26]
           of byte = (
[T14]   24:                                30,48,46,32,
           18,33,34,35,
[W04]   25:                                23,36,37,38,
           50,49,24,25,
[XC5]   26:                                16,19,31,20,
           22,47,17,45,21,44);
[M95]   27:                           s1 : set of byte =
            [
[T14]   28:                                30,48,46,32,
           18,33,34,35,
[103]   29:                                23,36,37,38,
           50];
[P95]   30:                           s2 : set of byte =
            [
[8C1]   31:                                49,24,25,
[NP5]   32:                                16,19,31,20,
           22,47,17,45,21,44];
[ZZ9]   33:    ERKL0 = 'Editierbare Zeichenkette';
[V3N]   34:    ERKL1 = '<ESC> Abbruch  Edit.: <'#27','#2
           6'> Bewegen, <Einfg/Ins> Insert/Overwrite,';
[USP]   35:    ERKL2 = 'Löschen: <Entf/Del> Unterm Curso
           r, <Rück./Back.> Vor Cursor, <F1> Alles';
[F9P]   36:    ERKL3 = '<F2> Alles ab Cursor, <F3> Alles
           vor Cursor, Wandel Groß-Kleinbuchstaben:';
[23P]   37:    ERKL4 = '<Ctrl> groß oder  <Alt> klein +
           <F1> Alles <F2> Vor Cursor <F3> Ab Cursor';
[6P6]   38:  var                ypos, pos : byte;
[D92]   39:                              ch : char;
[R83]   40:                               i : integer;
[LM4]   41:                         Cstr255 : tstr255;
[KG5]   42:                          str255 : string[255];
[L24]   43:                            str1 : string[1];
[Y03]   44:                        trotzdem,
[X09]   45:          gefunden, raus, ins : boolean;
[T0A]   46:                          merke : array[0..13]
           of string[65];
[EV7]   47:                           hilf : array[1..6,1.
           .150] of byte;
[000]   48:
[KDM]   49:   procedure wandeln( var st : tstr255; anf,
           anz : byte; gross : boolean );
[W63]   50:     var i : byte;
[BH1]   51:     begin
[VA6]   52:      for i := anf to anz do
[494]   53:       if gross then
[3C7]   54:        if st[ i ] in ['a'..'z'] then
[KF8]   55:         st[ i ] := chr( ord( st[ i ] ) - 32
           )
[A11]   56:        else
[UU4]   57:         case st[ i ] of
[N84]   58:           #132 : st[ i ] := #142;
[V94]   59:           #148 : st[ i ] := #153;
[Z94]   60:           #129 : st[ i ] := #154;
[DL0]   61:         end
[A11]   62:       else
[M87]   63:        if st[i ] in ['A'..'Z'] then
[KF8]   64:         st[ i ] := chr( ord( st[ i ] ) + 32
           )
[A11]   65:        else
[UU4]   66:         case st[ i ] of
[L84]   67:           #142 : st[ i ] := #132;
[3A4]   68:           #153 : st[ i ] := #148;
[3A4]   69:           #154 : st[ i ] := #129;
[TU0]   70:         end;
[YSA]   71:      moveStr( x1, y1 + pred( ypos ) , str25
           5 );
[TU0]   72:     end;
[000]   73:
[WKG]   74:   procedure loeschChr( var st : tstr255; an
           f, lanz : byte );
[W63]   75:    var i : byte;
[BH1]   76:    begin
[3Y6]   77:     delete( str255, anf, lanz );
[B37]   78:     if lanz = 1 then begin
[SS7]   79:      i := x1 + ( length( str255 ) );
[SM8]   80:      moveChr( i, y1 + pred( ypos ) , ' ' )
           ;
[DL0]   81:     end
[A11]   82:     else
[N2B]   83:      MoveStr( x1, y1 + pred( ypos ) , Cstr
           255 );
[YRA]   84:     MoveStr( x1, y1 + pred( ypos ) , str25
           5 );
[HBB]   85:     gotoxy( x1 + pred( anf ), y1 + pred( y
           pos ) );
[TU0]   86:    end;
[000]   87:
[9G9]   88:   procedure suchen( art : byte );
[L65]   89:    var i, j, h, k : byte;
[HW4]   90:                  su : set of byte;
[BH1]   91:    begin
[623]   92:     case art of
[882]   93:      3 : j := 120;
[B12]   94:      4 : j := 59;
[312]   95:      5 : j := 84;
[612]   96:      6 : j := 94;
[P82]   97:      7 : j := 104;
[TU0]   98:     end;
[C15]   99:     if art in [1,2] then
[SB1]  100:      k := 13
[A11]  101:     else
[3B1]  102:      k := 10;
[KE9]  103:     fillchar(merke,sizeof(merke),#32);
[454]  104:     for i := 0 to 13 do
[VC4]  105:      merke[i][0] := #0;
[ED1]  106:     i := 1;
[5S4]  107:     if art = 1 then
```

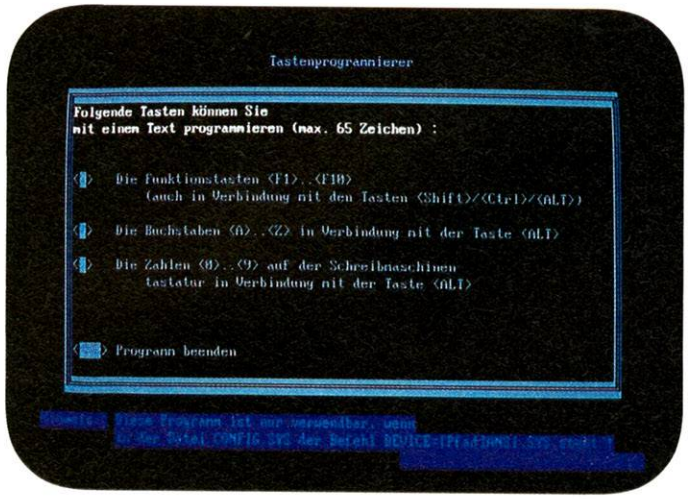


Bild 9. Das Menü des Programmes »Tasten«

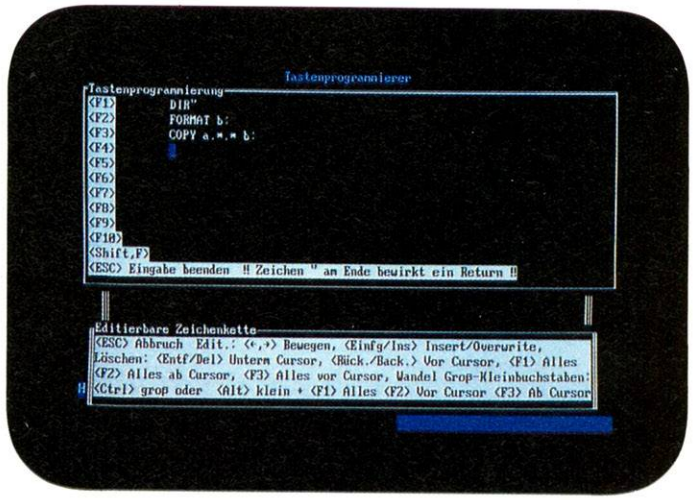


Bild 10. Die Belegung der Tasten geschieht sehr übersichtlich

```
[VX1]  108:         su := s1
[A11]  109:       else
[7S4]  110:        if art = 2 then
[VX1]  111:         su := s2
[A11]  112:        else
[N15]  113:         su := [j..pred(j+k)];
[T14]  114:       attribut := 7;
[6W6]  115:       if TextAnzahl > 0 then
[X52]  116:        repeat
[1PA]  117:         if Texte[i].taste in su then begin
[OT6]  118:          if art in [1,2] then begin
[9D1]  119:           j := 0;
[X52]  120:           repeat
[D43]  121:            j := succ(j);
[8A7]  122:           until Texte[i].taste = t[j];
[7S4]  123:           if art = 2 then
[F22]  124:            j := j - 13;
[1AA]  125:           MoveStr(x1,y1+pred(j),Texte[i].tex
       t);
[4V7]  126:           merke[j] := Texte[i].text;
[DLO]  127:          end
[YS2]  128:          else begin
[DSD]  129:           MoveStr(x1,y1+Texte[i].taste-j,Tex
       te[i].text);
[Y8F]  130:           merke[succ(Texte[i].taste-j)] := T
       exte[i].text;
[DLO]  131:          end
[TUO]  132:         end;
[943]  133:         i := succ(i);
[CT6]  134:        until ( i > TextAnzahl );
[TUO]  135:      end;
[OOO]  136:
[SDN]  137:   procedure finden( taste : byte; var i : i
       nteger; var gef : boolean );
[BH1]  138:     begin
[ED1]  139:      i := 1;
[OX2]  140:      gef := false;
[6W6]  141:      if TextAnzahl > 0 then
[X52]  142:       repeat
[YT9]  143:        if Texte[i].taste = taste then
[Y62]  144:         gef := true
[A11]  145:        else
[943]  146:         i := succ(i);
[SB8]  147:       until (i > TextAnzahl) or gef;
[TUO]  148:     end;
[OOO]  149:
[BH1]  150:     begin
[ZX8]  151:      MakeWindow(4,112,7,32, 2,19,75,6,hilf);
[T14]  152:      attribut := 7;
[T13]  153:      SetAdr(3,19);
[R09]  154:      SchreibeSatz(ERKL0);      NextLine;
[X84]  155:      Attribut := 112;
[T09]  156:      SchreibeSatz(ERKL1);      NextLine;
[V09]  157:      SchreibeSatz(ERKL2);      NextLine;
[X09]  158:      SchreibeSatz(ERKL3);      NextLine;
[EK5]  159:      SchreibeSatz(ERKL4);
[G93]  160:      Cursor(1,13);
[KP1]  161:      pos := 1;
[062]  162:      ypos := 1;
[GE5]  163:      trotzdem := false;
[C43]  164:      suchen(art);
[W86]  165:      str255 := merke[ypos];
[5G2]  166:      ins := true;
[TW3]  167:      raus := false;
[Z84]  168:      gotoxy( x1, y1 );
[KT6]  169:      Cstr255[0] := chr( anzahl );
[P7A]  170:      fillchar( Cstr255[1], anzahl, chr(32) )
       ;
[324]  171:      attribut := 9;
```

```
[157]  172:      moveStr( x1, y1, Cstr255 );
[AU6]  173:      moveStr( x1, y1, str255 );
[X52]  174:      repeat
[862]  175:       read2( ch );
[S95]  176:       if Nurgross then
[H84]  177:        ch := upcase( ch );
[UY5]  178:       if ch = #0 then begin
[862]  179:        read2( ch );
[RS2]  180:        case ch of
[R52]  181:          #71 : begin
[5L7]  182:                 gotoxy( x1, y1 + pred( ypos
       ) );
[KP1]  183:                 pos := 1;
[TUO]  184:                end;
[F62]  185:          #79 : begin
[P2D]  186:                 gotoxy( x1 + length(str255),
        y1 + pred( ypos ) );
[BM7]  187:                 pos := succ( length(str255)
       );
[TUO]  188:                end;
[207]  189:          #94 : if not Nurgross then
[M8C]  190:                 wandeln( str255, 1, length(s
       tr255), true );
[P17]  191:          #104: if not Nurgross then
[K8C]  192:                 wandeln( str255, 1, length(s
       tr255), false );
[K52]  193:          #60 : begin
[P9E]  194:                 loeschChr( str255, pos, leng
       th(str255) - pred( pos ) );
[US6]  195:                 if length(str255) = 0 then
[VG5]  196:                  trotzdem := true;
[TUO]  197:                end;
[N52]  198:          #61 : begin
[C17]  199:                 loeschChr( str255, 1, pos );
[KP1]  200:                 pos := 1;
[9B8]  201:                 gotoxy( x1 , y1 + pred( ypos
        ) );
[US6]  202:                 if length(str255) = 0 then
[VG5]  203:                  trotzdem := true;
[TUO]  204:                end;
[B62]  205:          #59 : begin
[NA2]  206:                 str255 := '';
[CT5]  207:                 merke[ypos] := str255;
[81A]  208:                 moveStr( x1, y1 + pred(ypos)
       , Cstr255 );
[KP1]  209:                 pos := 1;
[WM7]  210:                 gotoxy( x1, y1 + pred(ypos))
       ;
[VG5]  211:                 trotzdem := true;
[TUO]  212:                end;
[807]  213:          #96 : if not Nurgross then
[38C]  214:                 wandeln( str255, pos, length
       (str255), true );
[Y17]  215:          #106: if not Nurgross then
[7LC]  216:                 wandeln( str255, pos, length
       (str255), false );
[507]  217:          #95 : if not Nurgross then
[A79]  218:                 wandeln( str255, 1, pos, tru
       e );
[U17]  219:          #105: if not Nurgross then
[RK9]  220:                 wandeln( str255, 1, pos, fal
       se );
[9B6]  221:          #75 : if pos > 1 then begin
[B84]  222:                 pos := pred( pos );
[CYE]  223:                 gotoxy( x1 + pred( pos ), y1
        + pred( ypos ) );     { links  }
[TUO]  224:                end;
[A6C]  225:          #77 : if  pos < succ( length(str255
       ) ) then begin
```

```
[6WC]  226:                  gotoxy( x1 + pos, y1 + pred(
       ypos ) );     { rechts }
[784]  227:                  pos := succ( pos );
[TUO]  228:                end;
[1GH]  229:          #80 : if (ypos < pred(yanz)) and (l
       ength(str255) = 0) then begin
[KP1]  230:                  pos := 1;
[T14]  231:                  attribut := 7;
[A5A]  232:                  MoveStr(x1,y1+pred(ypos),Cst
       r255);
[LV9]  233:                  MoveStr(x1,y1+pred(ypos),str
       255);
[324]  234:                  attribut := 9;
[F25]  235:                  ypos := succ(ypos);
[W86]  236:                  str255 := merke[ypos];
[CP9]  237:                  MoveStr(x1,y1 + pred( ypos )
       , Cstr255);
[PH9]  238:                  MoveStr(x1,y1 + pred( ypos )
       , str255);
[TH7]  239:                  gotoxy( x1,y1 + pred( ypos )
       );
[DLO]  240:                end
[A11]  241:                else
[5L8]  242:                 if  length(str255) = 0 then
       begin
[KP1]  243:                  pos := 1;
[T14]  244:                  attribut := 7;
[32A]  245:                  MoveStr(x1, y1 + pred(ypos)
       , Cstr255);
[SZ9]  246:                  MoveStr(x1, y1 + pred(ypos)
       , str255);
[324]  247:                  attribut := 9;
[062]  248:                  ypos := 1;
[W86]  249:                  str255 := merke[ypos];
[2P6]  250:                  MoveStr(x1, y1, Cstr255 );
[A86]  251:                  MoveStr(x1, y1, str255);
[Z84]  252:                  gotoxy( x1, y1 );
[TUO]  253:                end;
[7VB]  254:          #82 : if ins then begin
                        { Einfg  }
[VY2]  255:                 ins := false;
[D93]  256:                 Cursor(5,12);
[DLO]  257:                end
[YS2]  258:                else begin
[5G2]  259:                 ins := true;
[G93]  260:                 Cursor(1,13);
[TUO]  261:                end;
[0Z7]  262:          #83 : if ( length(str255) > 0 ) and
[NBE]  263:                   ( length(str255) < succ( a
       nzahl ) ) then begin
[BK7]  264:                 loeschChr( str255, pos, 1 );
[US6]  265:                 if length(str255) = 0 then
[VG5]  266:                  trotzdem := true;
[TUO]  267:                end;
[TUO]  268:        end;
[DLO]  269:       end
[YS2]  270:       else begin
[RS2]  271:        case ch of
[EA6]  272:          #8 : if pos > 1 then begin
[B84]  273:                pos := pred( pos );
[BK7]  274:                loeschChr( str255, pos, 1 );
[TUO]  275:               end;
[8Z2]  276:         #27 : art := 0;
[CU9]  277:         #13 : raus := true
                   { Return }
[A11]  278:        else
[1E6]  279:        if ch in bereich then
[OMB]  280:         if (ypos <> pred(yanz)) or (art = 3)
       then
[MV4]  281:          if ins then begin
[CEB]  282:           if length(str255) < anzahl then be
       gin
[082]  283:            str1 := ch;
[1W6]  284:            insert( str1, str255, pos );
[YSA]  285:            moveStr( x1, y1 + pred( ypos ) ,
       str255 );
[UA9]  286:            gotoxy( x1 + pos, y1 + pred( ypos
       ) );
[784]  287:            pos := succ( pos );
[TUO]  288:           end;
[DLO]  289:          end
[YS2]  290:          else begin
[HAA]  291:           if pos > length( str255 ) then beg
       in
[CEB]  292:            if length(str255) < anzahl then b
       egin
[B75]  293:             str255 := str255 + ch;
[MA5]  294:             i := x1 + pred( pos );
[BL8]  295:             moveChr( i, y1 + pred( ypos ) ,
       ch );
[UA9]  296:             gotoxy( x1 + pos, y1 + pred( ypo
       s ) );
[784]  297:             pos := succ( pos );
[TUO]  298:            end;
[DLO]  299:           end
[YS2]  300:           else begin
[NM4]  301:            str255[ pos ] := ch;
[MA5]  302:            i := x1 + pred( pos );
```

```
[BL8]  303:            moveChr( i, y1 + pred( ypos ) , c
       h );
[784]  304:            pos := succ( pos );
[RF9]  305:            gotoxy( succ( i ), y1  + pred( yp
       os ) );
[TUO]  306:           end;
[TUO]  307:          end;
[TUO]  308:        end;
[TUO]  309:       end;
[4BA]  310:       if raus or trotzdem then begin
[GE5]  311:        trotzdem := false;
[F3T]  312: { trotzdem : falls zeile geloscht und mit c
       ursortasten weitergegangen wird !!! }
[623]  313:       case art of
[A38]  314:           1 : if ypos = pred(yanz) then
[HM1]  315:                art := 2
[A11]  316:               else
[TW3]  317:                raus := false;
[B38]  318:           2 : if ypos = pred(yanz) then
[HM1]  319:                art := 1
[A11]  320:               else
[TW3]  321:                raus := false;
[KT4]  322:           3 : raus := false;
[DK8]  323:       4..6 : if ypos = pred(yanz) then
[KA4]  324:                art := succ(art)
[A11]  325:               else
[TW3]  326:                raus := false;
[G38]  327:           7 : if ypos = pred(yanz) then
[HM1]  328:                art := 4
[A11]  329:               else
[TW3]  330:                raus := false;
[TUO]  331:       end;
[8RA]  332:       if (art <> 0) and not raus then begi
       n
[CT5]  333:        merke[ypos] := str255;
[T14]  334:        attribut := 7;
[80A]  335:        MoveStr( x1, y1 + pred(ypos), Cstr2
       55 );
[A9A]  336:        MoveStr( x1, y1 + pred(ypos), str25
       5 );
[623]  337:         case art of
[ZS3]  338:          3 : pos := 120;
[SF3]  339:          4 : pos := 59;
[CF3]  340:          5 : pos := 84;
[HF3]  341:          6 : pos := 94;
[HT3]  342:          7 : pos := 104;
[TUO]  343:         end;
[5S4]  344:         if art = 1 then
[XU3]  345:          pos := t[ypos]
[A11]  346:         else
[7S4]  347:          if art = 2 then
[5Z3]  348:           pos := t[ypos+13]
[A11]  349:          else
[H96]  350:           pos := pos + pred(ypos);
[7O7]  351:        finden(pos,i,gefunden);
[BY7]  352:        if not gefunden then begin
[XP9]  353:         TextAnzahl := succ(TextAnzahl);
[B3A]  354:         Texte[TextAnzahl].text := str255;
[TNA]  355:         Texte[TextAnzahl].taste := pos;
[DLO]  356:        end
[A11]  357:        else
[4Z6]  358:         Texte[i].text := str255;
[VZ7]  359:        if ypos <> pred(yanz) then
[YW4]  360:         ypos := succ(ypos)
[A11]  361:        else
[062]  362:         ypos := 1;
[OW7]  363:        gotoxy( x1, y1 + pred(ypos) );
[OMB]  364:        if (ypos <> pred(yanz)) or (art = 3
       ) then
[UX5]  365:         str255 := merke[ypos]
[A11]  366:        else
[NA2]  367:         str255 := '';
[324]  368:        attribut := 9;
[A5A]  369:        MoveStr(x1,y1+pred(ypos),Cstr255);
[LV9]  370:        MoveStr(x1,y1+pred(ypos),str255);
[KP1]  371:        pos := 1;
[TUO]  372:       end;
[TUO]  373:      end;
[1V5]  374:    until raus or (art = 0);
[SL3]  375:    cursor(32,32);
[NG8]  376:    RestoreWindow( 2,19,75,6,hilf);
[TUO]  377:   end;
[000]  378:
[L7A]  379:  procedure wahl( var art : byte );
[VMA]  380:   const  bereich : tsetofchar = [#32..#255]
       ;
[6B5]  381:   var       scan : byte;
[8S7]  382:               h1 : array[5..24,1..158] of b
       yte;
[WZ3]  383:               a, i : integer;
[BH1]  384:    begin
[623]  385:     case art of
[2L2]  386:      1..2 : a := 17;
[3Z1]  387:         3 : a := 13;
[7L2]  388:      4..7 : a := 14;
[TUO]  389:     end;
[LE7]  390:     MakeWindow(3,7,7,32,1,3,79,a,h1);
[T14]  391:     attribut := 7;
```

```
[K6B]  392:      MoveStr(2,3,'Tastenprogrammierung');
[X94]  393:      attribut := 112;
[9ET]  394:      MoveStr(2,succ(a),'<ESC> Eingabe beende
       n  '#19' Zeichen " am Ende bewirkt ein Retur
        n '#19);
[623]  395:      case art of
[692]  396:       1 : begin
[5K4]  397:            for i := 65 to 77 do
[EP7]  398:             MoveStr(2,i-61,'<ALT,'+chr(i)+'>
       ');
[FW5]  399:            MoveStr(2,17,'<N>..<Z>');
[TU0]  400:           end;
[792]  401:       2 : begin
[9K4]  402:            for i := 78 to 90 do
[5R7]  403:             MoveStr(2,i-74,'<ALT,'+chr(i)+'>
       ');
[EU5]  404:            MoveStr(2,17,'<A>..<M>');
[TU0]  405:           end;
[892]  406:       3 : begin
[DK4]  407:            for i := 49 to 57 do
[WP7]  408:             MoveStr(2,i-45,'<ALT,'+chr(i)+'>
       ');
[0L5]  409:            MoveStr(2,13,'<ALT,0>');
[TU0]  410:           end;
[992]  411:       4 : begin
[DK4]  412:            for i := 49 to 57 do
[208]  413:             MoveStr(2,i-45,'<F'+chr(i)+'>');
[Z75]  414:            MoveStr(2,13,'<F10>');
[UF6]  415:            MoveStr(2,14,'<Shift,F>');
[TU0]  416:           end;
[A92]  417:       5 : begin
[DK4]  418:            for i := 49 to 57 do
[U3A]  419:             MoveStr(2,i-45,'<Shift,F'+chr(i)
       +'>');
[FR6]  420:            MoveStr(2,13,'<Shift,F10>');
[R26]  421:            MoveStr(2,14,'<Ctrl,F>');
[TU0]  422:           end;
[B92]  423:       6 : begin
[DK4]  424:            for i := 49 to 57 do
[AD9]  425:             MoveStr(2,i-45,'<Ctrl,F'+chr(i)+
       '>');
[L96]  426:            MoveStr(2,13,'<Ctrl,F10>');
[WN5]  427:            MoveStr(2,14,'<ALT,F>');
[TU0]  428:           end;
[C92]  429:       7 : begin
[DK4]  430:            for i := 49 to 57 do
[MA8]  431:             MoveStr(2,i-45,'<ALT,F'+chr(i)+'
       >');
[ZY5]  432:            MoveStr(2,13,'<ALT,F10>');
[OR4]  433:            MoveStr(2,14,'<F>');
[TU0]  434:           end;
[W3C]  435:     end;          {anz,yanz,x,y,art,NurGroß
       }
[T14]  436:     attribut := 7;
[GRB]  437:     reading(bereich,65,a-2,14,4,art,false);
[427]  438:     RestoreWindow(1,3,79,a,h1);
[TU0]  439:    end;
[000]  440:
[7T5]  441:  procedure haupt;
[BW3]  442:   var ch : char;
[PF2]  443:       a : byte;
[000]  444:
[DP5]  445:   procedure lesen;
[7Y5]  446:    var     f : file of tr;
[R83]  447:            i : integer;
[BH1]  448:      begin
[S66]  449:       assign(f,'TAST76SP.TPG');
[2L2]  450:       reset(f);
[ER1]  451:       INT24F;
[W27]  452:       if ioresult = 0 then begin
[8D1]  453:        i := 0;
[ACC]  454:        while (ioresult = 0) and not eof(f) d
       o begin
[943]  455:         i := succ(i);
[RW4]  456:         read(f,Texte[i]);
[TU0]  457:        end;
[XP4]  458:        TextAnzahl := i;
[W27]  459:        if ioresult = 0 then begin
[2K2]  460:         close(f);
[ER1]  461:         INT24F;
[DL0]  462:        end
[A11]  463:        else
[WH2]  464:         signal;
[TU0]  465:       end;
[TU0]  466:      end;
[000]  467:
[MG6]  468:   procedure schreiben;
[7Y5]  469:    var     f : file of tr;
[R83]  470:            i : integer;
[BH1]  471:      begin
[C48]  472:       if TextAnzahl > 0 then begin
[S66]  473:        assign(f,'TAST76SP.TPG');
[UA3]  474:        rewrite(f);
[ER1]  475:        INT24F;
[W27]  476:        if ioresult = 0 then begin
[ED1]  477:         i := 1;
[AZD]  478:         while (ioresult = 0) and (i < TextAn
       zahl) do begin
[V47]  479:          if Texte[i].text <> '' then
[NEH]  480:           write(f,Texte[i]); {Hier nicht jed
       esmal INT24F aufrufen}
[943]  481:          i := succ(i);
[TU0]  482:         end;
[W27]  483:         if ioresult = 0 then begin
[2K2]  484:          close(f);
[ER1]  485:          INT24F;
[DL0]  486:         end
[A11]  487:         else
[WH2]  488:          signal;
[DL0]  489:        end
[A11]  490:        else
[WH2]  491:         signal;
[TU0]  492:       end;
[TU0]  493:      end;
[000]  494:
[BH1]  495:    begin
[NM2]  496:     clrscr;
[324]  497:     attribut := 9;
[FWA]  498:     MoveStr(31,2,'Tastenprogrammierer');
[0D4]  499:     attribut := 17;
[E4B]  500:     MoveStr(48,25,'(c) A. Baque  Bonn, den
       16.8.87');
[CD4]  501:     attribut := 19;
[2N6]  502:     MoveStr(1,23,'Hinweis :');
[CPG]  503:     MoveStr(11,23,'Diese Programm ist nur v
       erwendbar, wenn');
[4BN]  504:     MoveStr(11,24,'in der Datei CONFIG.SYS
       der Befehl DEVICE=[Pfad]ANSI.SYS steht !');
[MC4]  505:     attribut := 15;
[DG7]  506:     MakeWindow(2,7,112,32,4,4,72,18,h1);
[VEC]  507:     MoveStr(5,5,'Folgende Tasten können Sie
       ');
[VVK]  508:     MoveStr(5,6,'mit einem Text programmier
       en (max. 65 Zeichen) :');
[T14]  509:     attribut := 7;
[TND]  510:     MoveStr(5,9,'< >   Die Funktionstasten
       <F1>..<F10>');
[SAN]  511:     MoveStr(15,10,'(auch in Verbindung mit
       den Tasten <Shift>/<Ctrl>/<ALT>)');
[G6P]  512:     MoveStr(5,12,'< >   Die Buchstaben <A>.
       .<Z> in Verbindung mit der Taste <ALT>');
[7VK]  513:     MoveStr(5,14,'< >   Die Zahlen <0>..<9>
        auf der Schreibmaschinen-');
[UXG]  514:     MoveStr(15,15,'tastatur in Verbindung m
       it der Taste <ALT>');
[MTA]  515:     MoveStr(5,19,'<   > Programm beenden');
[N94]  516:     attribut := 120;
[WO4]  517:     MoveChr(6,9,'1');
[A54]  518:     MoveChr(6,12,'2');
[T54]  519:     MoveChr(6,14,'3');
[WT4]  520:     MoveStr(6,19,'ESC');
[F56]  521:     for a := 1 to 76 do begin
[SM5]  522:      Texte[a].taste := 0;
[AH6]  523:      Texte[a].text[0] := #0;
[TU0]  524:     end;
[0D1]  525:     a := 0;
[EA4]  526:     TextAnzahl := 0;
[BX1]  527:     lesen;
[X52]  528:     repeat
[D92]  529:      read2(ch);
[UY5]  530:      if ch = #0 then begin
[D92]  531:       read2(ch);
[DL0]  532:      end
[YS2]  533:      else begin
[RS2]  534:       case ch of
[UT7]  535:        #49 : begin a := 4; wahl(a); end;
[VS7]  536:        #50 : begin a := 1; wahl(a); end;
[2T7]  537:        #51 : begin a := 3; wahl(a); end;
[TU0]  538:       end;
[6L7]  539:       if (a <> 0) and (ch <> #27) then
[X52]  540:        repeat
[E22]  541:         wahl(a);
[DY2]  542:        until a = 0;
[TU0]  543:      end;
[9U3]  544:     until ch = #27;
[U13]  545:     schreiben;
[TU0]  546:    end;
[000]  547:
[AFC]  548:  begin   {CBreak:=false bringt hier nichts}
[FTA]  549:   ctrlbreak(true,ctrlbreakstatus);
[MP7]  550:   ctrlbreak(false,trueB);
[7N5]  551:   CursorFrage(0,ca,ce);
[SK3]  552:   Cursor(32,32);
[DK2]  553:   screen;
[S19]  554:   move(mem[screenaddr:0],h[1],4000);
[0B3]  555:   wx := wherex;
[6B3]  556:   wy := wherey;
[WY1]  557:   Int24S;
[FZ1]  558:   haupt;
[PY1]  559:   Int24R;
[YL9]  560:   move(h[1],mem[screenaddr:0],4000);
[B74]  561:   gotoxy(wx,wy);
[3Z3]  562:   Cursor(ca,ce);
[99B]  563:   CtrlBreak(false,CtrlBreakStatus);
[5T0]  564:  end.
```

Listing 17. Quellcode für »Tasten« zur Programmierung von Tastenbelegungen

```
[U8G]   0: {tastbat.pas 17.8.87 (Geändert für DOS inte
          rnational 8.1.88)}
[000]   1:
[G65]   2: {$I-,R-,D-,V-,K-,C-,U-}
[000]   3:
[WP3]   4: {$I LIBRARY.INC}
[000]   5:
[THE]   6: {Hinweis zum Abtippen: Vieles ähnlich mit T
          ASTEN}
[000]   7:
[E67]   8: type                    tr = record
[FG4]   9:                                text : string[6
          5];
[TG3]  10:                               taste : byte;
[TU0]  11:                             end;
[GY7]  12:                    tTexte = array[1..76] of t
          r;
[4G7]  13: var                  regs : tregs;
[PB6]  14:                TextAnzahl : integer;
[FS4]  15:                     Texte : tTexte;
[000]  16:
[7T5]  17: procedure haupt;
[BW3]  18:  var ch : char;
[000]  19:
[GR5]  20:   procedure taste;
[E43]  21:     var a : char;
[BH1]  22:      begin
[WY1]  23:       read2( a );
[OG3]  24:       if a = #0 then
[WY1]  25:        read2( a );
[TU0]  26:      end;
[000]  27:
[DP5]  28:    procedure lesen;
[7Y5]  29:     var     f : file of tr;
[R83]  30:             i : integer;
[BH1]  31:      begin
[EA4]  32:       TextAnzahl := 0;
[S66]  33:       assign(f,'TAST76SP.TPG');
[2L2]  34:       reset(f);
[W27]  35:       if ioresult = 0 then begin
[8D1]  36:        i := 0;
[ACC]  37:        while (ioresult = 0) and not eof(f) d
          o begin
[943]  38:         i := succ(i);
[RW4]  39:         read(f,Texte[i]);
[TU0]  40:        end;
[XP4]  41:        TextAnzahl := i;
[8B5]  42:        if ioresult = 0 then
[7G2]  43:         close(f)
[A11]  44:        else
[WH2]  45:         signal;
[TU0]  46:       end;
[TU0]  47:      end;
[000]  48:
[K8A]  49:    procedure frage( var ch : char );
[BH1]  50:      begin
[X52]  51:       repeat
[D92]  52:        read2(ch);
[YE4]  53:        ch := upcase(ch);
[UY5]  54:        if ch = #0 then begin
[D92]  55:         read2(ch);
[9G1]  56:         ch := #32;
[TU0]  57:        end;
[8M5]  58:       until ch in ['J','N'];
[TU0]  59:      end;
[000]  60:
[SM5]  61:    procedure patch;
[6MB]  62:     const fest : string[13] = '\TASTEN.BAT'#
          13#10;
[6F5]  63:     type     tf = Text[256];
[4D3]  64:     var       f : tf;
[BH1]  65:      begin
[ZA3]  66:       SetAdr(10,18);
[HTL]  67:       SchreibeSatz('Die AUTOEXEC.BAT wird im
           Hauptverzeichnis gesucht.');
[ZV2]  68:       NextLine;
[1CK]  69:       SchreibeSatz('bzw. sonst dort, wohin d
          er Befehl PATH zeigt.');
[ZV2]  70:       NextLine;
[N5K]  71:       SchreibeSatz('Der Befehl \TASTEN.BAT w
          ird als letzter Befehl');
[ZV2]  72:       NextLine;
[2UF]  73:       SchreibeSatz('in die AUTOEXEC.BAT-Date
          i geschrieben !');
[DL6]  74:       assign(f,'\AUTOEXEC.BAT');
[UH2]  75:       Append(f);
[W27]  76:       if ioresult = 0 then begin
[3C5]  77:        writeln(f,fest);
[PV5]  78:        if ioresult <> 0 then
[Z42]  79:         signal
[A11]  80:        else
[2K2]  81:         close(f);
[DL0]  82:       end
[A11]  83:       else
[WH2]  84:        signal;
[DL8]  85:       MoveStr(1,22,'Taste drücken !');
[UX1]  86:       taste;
```

```
[CW5]  87:       MoveStr(1,22,'                ');
[TU0]  88:      end;
[000]  89:
[4V5]  90:    procedure wandeln;
[B5A]  91:     const    r : string[12] = 'PROMPT $e[0;'
          ;
[EY7]  92:              r1 : string[11] = 'PROMPT $n$g';
[FG1]  93:     var      i,
[365]  94:          laenge : integer;
[HL3]  95:               f : Text[512];
[PF2]  96:               a : byte;
[V94]  97:             st5 : string[5];
[F94]  98:            fehl : boolean;
[5Z3]  99:          str255 : tstr255;
[000] 100:
[20F] 101:      procedure sende( st : tstr255; var fehl
          : boolean );
[BH1] 102:        begin
[ZE4] 103:         Writeln(f,st);
[B76] 104:         fehl := (ioresult <> 0);
[TU0] 105:        end;
[000] 106:
[BH1] 107:      begin
[N36] 108:       assign(f,'\TASTEN.BAT');
[UA3] 109:       rewrite(f);
[W27] 110:       if ioresult = 0 then begin
[FU2] 111:        laenge := 0;
[8D1] 112:        i := 0;
[X52] 113:        repeat
[55D] 114:         laenge := laenge + length(Texte[i].t
          ext);
[943] 115:         i := succ(i);
[CO7] 116:        until i >= TextAnzahl;
[1U9] 117:        laenge := laenge + 256;  { Reserve }
[SM4] 118:        str(laenge,st5);
[8D1] 119:        i := 0;
[OW4] 120:        if not fehl then
[X52] 121:         repeat
[943] 122:          i := succ(i);
[T67] 123:          str(Texte[i].taste,st5);
[LL4] 124:          str255 := r + st5 + ';"';
[BAF] 125:          if Texte[i].text[length(Texte[i].te
          xt)] <> '"' then
[DFA] 126:           str255 := str255 + Texte[i].text +
          '"p'
[YS2] 127:          else begin
[AUD] 128:           delete( Texte[i].text, length(Text
          e[i].text), 1 );
[4LA] 129:           str255 := str255 + Texte[i].text +
          '";13p';
[TU0] 130:          end;
[VN4] 131:          sende( str255, fehl );
[029] 132:         until fehl or (i >= TextAnzahl);
[OW4] 133:        if not fehl then
[XK4] 134:         sende( r1, fehl );
[2K2] 135:        close(f);
[ZA9] 136:        if (ioresult <> 0) or fehl then
[WH2] 137:         signal;
[DL0] 138:       end
[A11] 139:       else
[WH2] 140:        signal;
[DL8] 141:       MoveStr(1,22,'Taste drücken !');
[UX1] 142:       taste;
[CW5] 143:       MoveStr(1,22,'                ');
[TU0] 144:      end;
[000] 145:
[VO6] 146:    procedure erzeuge;
[BH1] 147:      begin
[BX1] 148:       lesen;
[6W6] 149:       if TextAnzahl > 0 then
[PW2] 150:        wandeln;
[TU0] 151:      end;
[000] 152:
[BH1] 153:    begin
[NM2] 154:     clrscr;
[324] 155:     attribut := 9;
[N5M] 156:     MoveStr(18,4,'Batch-Datei-Erzeugung zur
          Tastenprogrammierung');
[OD4] 157:     attribut := 17;
[F4B] 158:     MoveStr(48,25,'(c) A. Baqué  Bonn, den 1
          7.8.87');
[OD4] 159:     attribut := 17;
[2N6] 160:     MoveStr(1,23,'Hinweis :');
[T14] 161:     attribut := 7;
[RVN] 162:     MoveStr(11,23,'Die erzeugte Batch-Datei
          "TASTEN.BAT" ist nur verwendbar.');
[8CR] 163:     MoveStr(5,24,'wenn in der Datei CONFIG.S
          YS der Befehl DEVICE=[Pfad]ANSI.SYS steht !'
          );
[G13] 164:     SetAdr(10,8);
[PRM] 165:     SchreibeSatz('Die Datei "TAST76SP.TPG" w
          ird im aktuellen Pfad benötigt.');
[ZV2] 166:     NextLine;
[8DR] 167:     SchreibeSatz('Anschliessend wird eine Ba
          tch-Datei mit dem Namen "TASTEN.BAT"');
[ZV2] 168:     NextLine;
[GHE] 169:     SchreibeSatz('im Hauptverzeichnis erzeug
          t.');
```

```
[ZV2]  170:     NextLine;
[3EN]  171:     SchreibeSatz('Diese kann dann beim Syste
          mstart (booten) ausgefuhrt');
[ZV2]  172:     NextLine;
[4ER]  173:     SchreibeSatz('werden, oder direkt in die
          AUTOEXEC.BAT-Datei übernommen werden.');
[MC4]  174:     attribut := 15;
[3EC]  175:     MoveStr(20,14,'Batch-Datei erzeugen ? [J
          /N] :');
[6K2]  176:     frage(ch);
[B86]  177:     if ch = 'J' then begin
[494]  178:      MoveChr(51,14,'J');
[1Y2]  179:      erzeuge;
[C48]  180:      if TextAnzahl > 0 then begin
[LA3]  181:       SetAdr(20,15);
[PUB]  182:       SchreibeSatz('Soll die Datei bei jedem
          ');
[ZV2]  183:       NextLine;
[D4L]  184:       SchreibeSatz('Systemstart (booten) aus
          geführt werden ? [J/N] :');
[6K2]  185:       frage(ch);
[V54]  186:       if ch = 'J' then
[4W1]  187:        patch;
[DL0]  188:      end
[A11]  189:      else
[WH2]  190:       signal;
[TU0]  191:    end;
[TU0]  192:   end;
[000]  193:
[AFC]  194:  begin   {CBreak:=false bringt hier nichts}
[FTA]  195:   ctrlbreak(true,ctrlbreakstatus);
[MP7]  196:   ctrlbreak(false,trueB);
[7N5]  197:   CursorFrage(0,ca,ce);
[043]  198:   wx := Wherex;
[613]  199:   WY := Wherey;
[SK3]  200:   Cursor(32,32);
[DK2]  201:   screen;
[8S8]  202:   move(mem[screenaddr:0],h,4000);
[WY1]  203:   Int24S;
[FZ1]  204:   haupt;
[PY1]  205:   Int24R;
[0L8]  206:   move(h,mem[screenaddr:0],4000);
[B74]  207:   gotoxy(wx,wy);
[3Z3]  208:   Cursor(ca,ce);
[99B]  209:   CtrlBreak(false,CtrlBreakStatus);
[5T0]  210:  end.
```

Listing 18. »Tastbat« wandelt die Tastaturbelegung in eine Stapeldatei um

```
[HYH]    0: {tastzeig.pas 17.8.87 (Geandert für DOS int
          ernational 8.1.88)}
[000]    1:
[A45]    2: {$I-,D-,R-,V-,K-,C-,U-}
[000]    3:
[59E]    4: {Hinweis zum Abtippen: Wie schon TASTBAT gi
          bt es
[P4A]    5:  auch hier viel ähnliches mit TASTEN
[4CC]    6:  Dieses Programm braucht keine LIBRARY !!!}
[000]    7:
[HTB]    8: const           tast : array[1..5] of strin
          g[8] = (
[9R0]    9:                        '<F',
[DZ2]   10:                        '<Shift.F',
[DD2]   11:                        '<Ctrl.F',
[MU1]   12:                        '<ALT.F',
[ZP0]   13:                        '<');
[DY6]   14:                 t : array[1..26] of byte =
          (
[T14]   15:                              30,48,46,32,18,3
          3,34,35,
[W04]   16:                              23,36,37,38,50,4
          9,24,25,
[XC5]   17:                              16,19,31,20,22,4
          7,17,45,21,44);
[U35]   18: type          tr = record
[FG4]   19:                      text : string[65];
[TG3]   20:                     taste : byte;
[TU0]   21:                    end;
[GY7]   22:           tTexte = array[1..76] of tr;
[5U2]   23: var      c, e, k,
[HN7]   24:     i, TextAnzahl : integer;
[FS4]   25:             Texte : tTexte;
[0B4]   26:              expr : string[4];
[TL4]   27:                 f : file of tr;
[KG5]   28:           str255 : string[255];
[9UA]   29:          attribut : byte absolute dseg:8;
[000]   30:
[BH1]   31: begin
[S66]   32:  assign(f,'TAST76SP.TPG');
[2L2]   33:  reset(f);
[W27]   34:  if ioresult = 0 then begin
```

```
[8D1]   35:   i := 0;
[ACC]   36:   while (ioresult = 0) and not eof(f) do be
          gin
[943]   37:    i := succ(i);
[RW4]   38:    read(f,Texte[i]);
[TU0]   39:   end;
[XP4]   40:   TextAnzahl := i;
[C48]   41:   if TextAnzahl > 0 then begin
[X94]   42:    attribut := 112;
[6Y8]   43:    write('Programmierte Tasten :');
[T14]   44:    attribut := 7;
[9Z2]   45:    writeln;
[XU2]   46:    write(':12);
[0D4]   47:    attribut := 17;
[Y6K]   48:    write('"#13" bedeutet, daß der Befehl di
          rekt ausgeführt wird !');
[T14]   49:    attribut := 7;
[SG4]   50:    writeln(^j^m);
[X79]   51:    for i := 1 to TextAnzahl do begin
[AS6]   52:     case Texte[i].taste of
[09C]   53:          59..68 : begin k := 1; e := Texte[i
          ].taste - 58; end;
[Y7C]   54:          84..93 : begin k := 2; e := Texte[i
          ].taste - 83; end;
[03D]   55:         94..103 : begin k := 3; e := Texte[i
          ].taste - 93; end;
[X2D]   56:        104..113 : begin k := 4; e := Texte[i
          ].taste - 103; end;
[5S3]   57:        120..129 : begin
[EE1]   58:                    k := 6;
[M78]   59:                    if Texte[i].taste <> 129
          then
[1Z5]   60:                     e := Texte[i].taste - 11
          9
[A11]   61:                   else
[4D1]   62:                    e := 0;
[TU0]   63:                  end;
[YS2]   64:       else begin
[GD1]   65:        k := 1;
[X52]   66:        repeat
[H43]   67:         k := succ(k);
[7PC]   68:        until (k = 27) or (Texte[i].taste = t
          [pred(k)]);
[4A4]   69:        expr := chr(k + 63);
[8E1]   70:        k := 5;
[TU0]   71:       end;
[TU0]   72:      end;
[TR3]   73:      if k <> 5 then
[GY2]   74:       str(e,expr);
[913]   75:      if k = 6 then
[B33]   76:       k := pred(k);
[024]   77:      expr := expr + '>';
[VTH]   78:      if Texte[i].text[length(Texte[i].text)]
          = '"' then begin
[AUD]   79:       delete( Texte[i].text, length(Texte[i]
          .text), 1 );
[PC8]   80:       str255 := Texte[i].text + '#13';
[DL0]   81:      end
[A11]   82:      else
[K77]   83:       str255 := Texte[i].text;
[MC4]   84:      attribut := 15;
[7T6]   85:      write(tast[k]+expr:11,' ');
[T14]   86:      attribut := 7;
[8P4]   87:      writeln(str255);
[TU0]   88:     end;
[DL0]   89:    end
[A11]   90:    else
[OM2]   91:     write(^g);
[DL0]   92:   end
[YS2]   93:   else begin
[OM2]   94:    write(^g);
[N94]   95:    attribut := 120;
[S9C]   96:    write('Datei "TAST76SP.TPG" nicht gefunde
          n !');
[T14]   97:    attribut := 7;
[9Z2]   98:    writeln;
[TU0]   99:   end;
[5T0]  100: end.
[000]  101:
```

Listing 19. Das Programm »TastZeig« zeigt die Tastaturbelegung an

Menügesteuerter Programmaufruf mit »Execute«

Dieses Programm dient dazu, andere Programme aufzurufen und anschließend wieder in das Execute-Programm zurückzukehren. Alle ausführbaren Programme werden in einem zweidimensionalen Pulldown-Menü dargestellt, lassen sich mit den Cursorsteuertasten anwählen und mit [Return] starten (Bild 11). Zudem werden alle Unterverzeichnisse und das übergeordnete Verzeichnis (soweit

vorhanden) als Menüpunkte angeboten. Die Bedienung ist somit denkbar einfach. Das Programm basiert auf der Directory-Routine »Sdir« und bedarf nur noch geringfügiger Ergänzungen. Haben Sie »Sdir« bereits abgetippt, so bereitet Ihnen dieses Programm kaum mehr Arbeit. Der Hauptteil von »Execute« ist eine kompakte Routine, die ohne »Inline« auskommt. Es kann jedenfalls nichts passieren, was zum Programmabsturz führen würde. Die Programme werden außerdem nicht über den Kommandoprozessor aufgerufen, es sei denn, Sie wählen direkt »command.com« als Datei. In diesem Fall ist Ihnen gestattet, einen Parameter zu übergeben (zum Beispiel »Dir /P«). Löschen Sie hierbei die vorgegebene Folge »/C«, so gelangen Sie nur mit dem Befehl »exit« zum Execute-Programm (Listing 20) zurück. Sollte beim Versuch, das Programm auszuführen, ein Fehler entstehen, so wird ein Window mit einer Fehlermeldung ausgegeben. »Execute« verfügt über dieselben Funktionen wie »Sdir«. Da das Programm im Gegensatz zu vielen anderen Programmen nicht über den Kommando-Prozessor läuft, muß dieser auch nicht im aktuellen Laufwerk stehen.
Beachten Sie bitte, daß das Programm nur als Com-Datei gestartet werden darf. Der dynamische Speicherbereich (»maximum free dynamic memory«, A-Option in Turbo Pascal) sollte klein gehalten werden (zum Beispiel $0500), damit Platz für die aufzurufenden Programme bleibt.

```
[375]    1: {$I-,K-,D-,R-,V-,C-,U-}
[3LF]    2: {ERASER 18.11.87 (Geändert für DOS internat
           ional 8.1.88)}
[000]    3:
[WP3]    4: {$I LIBRARY.INC}
[R64]    5: {$I WINDOW.INC}
[XU3]    6: {$I LIBRARYE.INC}
[4P3]    7: {$I TOOLS.INC}
[TT5]    8: procedure reading;
[VC8]    9:  type                    tstr80 = string[80];
[215]   10:  var           BFrage, BVerz,
[P55]   11:                      fehl, io : boolean;
[FZ3]   12:                         h, d : integer;
[A64]   13:                   drive, y, a : byte;
[C84]   14:                datei1, datei4,
[3A5]   15:                   datei5, pfad : tstr80;
[D92]   16:                             ch : char;
[PA4]   17:                             s4 : string[4];
[TD6]   18:                             as : array[1..43]
          of byte;
[000]   19:
[000]   20:
[WHP]   21:  procedure dateiSuchen( ersten : boolean; v
          ar a : byte; var dateiname : tstr80;
[886]   22:                             var fehler : boolea
          n);
[Z65]   23:   var regs : tregs;
[BH1]   24:    begin
[Y36]   25:     if ersten then begin
[375]   26:      with regs do begin
[AM7]   27:       dateiname := dateiname + #0;
[XF1]   28:       Ah := $1A;
[1Y3]   29:       DS := Seg(as[1]);
[VX3]   30:       DX := Ofs(as[1]);
[W33]   31:       MSDos(regs);
[1G1]   32:       Ah := $4E;
[CE1]   33:       CX := 30;
[8B6]   34:       DS := Seg(dateiname[1]);
[2B6]   35:       DX := Ofs(dateiname[1]);
[WW2]   36:       MSDos(Regs);
[TU0]   37:      end;
[DL0]   38:     end
[YS2]   39:     else begin
[A14]   40:      regs.ah := $4F;
[W33]   41:      MSDos(regs);
[TU0]   42:     end;
[G56]   43:     fehler := regs.al <> 0;
[ED7]   44:     if not Fehler then begin
[E15]   45:      dateiname[0] := #13;
[P58]   46:      move(as[31],dateiname[1],13);
[G3D]   47:      dateiname[0] := chr( pred(pos(#0,datei
          name)) );
[7R2]   48:       a := as[22];
[TU0]   49:     end;
[TU0]   50:    end;
[000]   51:
[SRN]   52:  procedure dateiloeschen( dateiname : tstr8
          0; var fehler : boolean );
[Z65]   53:   var regs : tregs;
[BH1]   54:    begin
[AM7]   55:     dateiname := dateiname + #0;
[375]   56:     with regs do begin
[DG1]   57:      ah := $41;
```

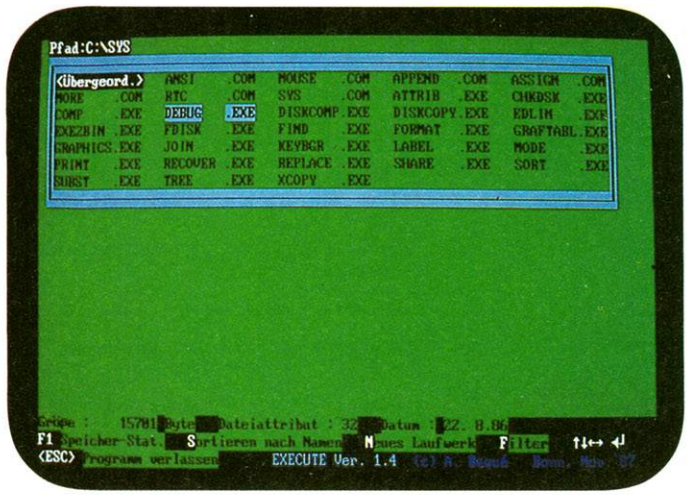


Bild 11. Mit »Execute« lassen sich Programme bequem über ein Pulldown-Menü starten

```
[8N6]   58:       ds := seg(dateiname[1]);
[2N6]   59:       dx := ofs(dateiname[1]);
[W93]   60:       msdos(regs);
[FE7]   61:       fehler := ( flags and 1 ) = 1;
[TU0]   62:      end;
[TU0]   63:     end;
[000]   64:
[VEA]   65:  procedure scroll( var y : byte );
[BH1]   66:    begin
[CB4]   67:     if y >= 25 then
[8H2]   68:      rollen1
[A11]   69:     else
[963]   70:      y := succ(y);
[893]   71:     gotoxy(1,y);
[TU0]   72:    end;
[000]   73:
[6S5]   74:  procedure infos;
[YBC]   75:   const s1 : string[14] = '[Laufw.:][Pfad';
[8V9]   76:         s2 : string[17] = '1]Dateiname.Erw.
          ';
[BH1]   77:    begin
[6N2]   78:     scroll(y);
[MC4]   79:     attribut := 15;
[7Y6]   80:     MoveStr(1,y,s1+']ERASER');
[T14]   81:     attribut := 7;
[XD5]   82:     MoveStr(25,y,s1+s2);
[6N2]   83:     scroll(y);
[6N2]   84:     scroll(y);
[7HH]   85:     MoveStr(1,y,s1+s2+': Dateien, die gelös
          cht werden sollen.');
[6N2]   86:     scroll(y);
[OAD]   87:     MoveStr(34,y,'Darf Wildcards enthalten
          (*,?).');
[MY5]   88:     scroll(y);scroll(y);
[UNL]   89:     MoveStr(1,y,'Fehlt Pfad1, so wird der a
          ktuelle Pfad angenommen !');
[MY5]   90:     scroll(y);scroll(y);
[65V]   91:     MoveStr(1,y,'Der Benutzer wird vor jede
          m Löschen gefragt, ob er dies wirklich will
           !');
[MY5]   92:     scroll(y);scroll(y);
[O2M]   93:     MoveStr(1,y,'Das Programm kann mit <Ctr
          l,C> abgebrochen werden.');
[6N2]   94:     scroll(y);
[TU0]   95:    end;
[000]   96:
[AL5]   97:  procedure Falsch;
[BH1]   98:    begin
[AGD]   99:     MoveStr(1,y,'Parameter oder -anzahl fal
          sch !');
[PZ1]  100:     int24R;
[99B]  101:     CtrlBreak(false,CtrlBreakStatus);
[5B1]  102:     halt;
[TU0]  103:    end;
[000]  104:
[EP4]  105:  procedure msg;
[BH1]  106:    begin
[6N2]  107:     scroll(y);
[FK2]  108:     str(d:4,s4);
[68B]  109:     MoveStr(1,y,s4+' Datei(en) gelöscht !')
          ;
[6N2]  110:     scroll(y);
[TU0]  111:    end;
[000]  112:
[RBA]  113:  procedure upper( var s : tstr255 );
[W63]  114:   var i : byte;
[BH1]  115:    begin
[HR6]  116:     for i := 1 to byte(s[0]) do
[5A5]  117:      s[i] := upcase(s[i]);
[TU0]  118:    end;
[000]  119:
```

```
[BH1]  120:    begin
[T14]  121:     attribut := 7;
[DK2]  122:     screen;
[3D1]  123:     d := 0;
[G13]  124:     y := wherey;
[DR6]  125:     if ParamCount = 0 then
[KL1]  126:      infos
[YS2]  127:     else begin
[20A]  128:      fillchar(datei1,sizeof(datei1),#32);
[NR6]  129:      if ParamCount > 1 then
[MG2]  130:       Falsch;
[SU5]  131:      datei1 := ParamStr(1);
[BW3]  132:      upper(datei1);
[W52]  133:      pfad := '';
[VE2]  134:      drive := 0;
[7ZB]  135:      if (datei1[2] = ':') and (datei1[0] > #
         0) then
[BL6]  136:       drive := ord(datei1[1])-64
[A11]  137:      else
[2E5]  138:       aktdrive(drive);
[9NL]  139:      if (datei1[0] = #0) or ((datei1[1] <> '
         \') and (datei1[3] <> '\')) then begin
[ZE5]  140:       aktdir(pfad,drive);
[TF4]  141:       if pfad <> '' then
[YA5]  142:        insert('\',pfad,1);
[RO9]  143:       if pfad[byte(pfad[0])] <> '\' then
[2V3]  144:        pfad := pfad + '\';
[876]  145:       insert(pfad,datei1,1);
[TUO]  146:      end;
[M75]  147:      h := length(datei1);
[4T4]  148:      datei4 := datei1;
[X52]  149:      repeat
[OT6]  150:       if datei4[h] <> '\' then
[MZ4]  151:        delete(datei4,h,1);
[Z23]  152:       h := pred(h);
[OH9]  153:      until (datei4[succ(h)] = '\') or (h = 0
         );
[1R9]  154:      dateiSuchen( true, a, datei1, fehl );
[VD7]  155:      while not fehl do begin
[3X7]  156:       if keypressed then begin
[D92]  157:        read2(ch);
[FD4]  158:        if ch = #0 then
[R52]  159:         read2(ch)
[A11]  160:        else
[OZ5]  161:        if ch = #3 then begin
[MVO]  162:         msg;
[WH2]  163:         signal;
[PZ1]  164:         int24R;
[99B]  165:         CtrlBreak(false,CtrlBreakStatus);
[5B1]  166:         halt;
[TUO]  167:        end;
[TUO]  168:       end;
[1DC]  169:       if boolean(a and 16) or boolean(a and
         8) or
[96E]  170:          boolean(a and 4) or boolean(a and 2
         ) then begin
[DLO]  171:       end
[YS2]  172:       else begin
[V47]  173:        datei5 := datei4+datei1;
[6N2]  174:        scroll(y);
[ERA]  175:        MoveStr(1,y,datei5+' löschen [J/N]');
[SA6]  176:        h := byte(datei5[0])+16;
[893]  177:        gotoxy(h,y);
[X52]  178:        repeat
[D92]  179:         read2(ch);
[YE4]  180:         ch := upcase(ch);
[FD4]  181:         if ch = #0 then
[D92]  182:          read2(ch);
[NH5]  183:        until ch in [#3,#74,#78];
[OZ5]  184:        if ch = #3 then begin
[MVO]  185:         msg;
[WH2]  186:         signal;
[PZ1]  187:         int24R;
[99B]  188:         CtrlBreak(false,CtrlBreakStatus);
[5B1]  189:         halt;
[TUO]  190:        end;
[RU4]  191:        MoveChr(h,y,ch);
[886]  192:        if ch = #74 then begin
[KO7]  193:         dateiLoeschen( datei5, io );
[8A4]  194:         if not io then
[N33]  195:          d := succ(d);
[TUO]  196:        end;
[TUO]  197:       end;
[YH9]  198:       dateiSuchen( false, a, datei1, fehl );
[TUO]  199:      end;
[MVO]  200:      msg;
[TUO]  201:     end;
[TUO]  202:    end;
[000]  203:
[BZC]  204:   begin    {CBreak:=false bringt hier nichts
         }
[FTA]  205:    ctrlbreak(true,ctrlbreakstatus);
[MP7]  206:    ctrlbreak(false,trueB);
[WZ1]  207:    int24S;
[1V2]  208:    reading;
[PZ1]  209:    int24R;
[99B]  210:    CtrlBreak(false,CtrlBreakStatus);
[5TO]  211:   end.
```

Listing 20. Das Turbo Pascal-Listing zu »Execute« zur menügesteuerten Programmausführung

Dateien löschen mit Sicherheitsabfrage

Das Programm »Eraser« (Listing 21) löscht mehrere Dateien, jedoch nicht ohne Sicherheitsfrage bei jedem einzelnen Directory-Eintrag. Auf diese Weise kommt endlich die Angabe »*.*« als Kriterium für den Löschvorgang zum Tragen. »Eraser« fragt dann vor jedem Löschversuch, ob die Datei auch wirklich aus dem Verzeichnis entfernt werden soll. Die Prozedur besitzt folgende Syntax:

```
Eraser [Laufw.:][Pfad1]Dateiname.Erw.
```

Der Dateiname darf natürlich Joker und Wildcards enthalten (»*«, »?«). Fehlt die Angabe »Pfad1«, so nimmt das Programm den aktuellen Pfad an. Die Prozedur ist gegen Datenverlust gesichert und kann mit [Ctrl-C] unterbrochen werden.

```
[P65]    0: {$I-,V-,C-,U-,K-,D-,R-}
[000]    1:
[7NH]    2: {EXECUTE.PAS Neu: 9.1.88 (Geändert für DOS
           international 9.1.88)}
[WBK]    3: {Nur als COM-Datei laufen lassen, aber mit
           runtergesetztem
[GEB]    4:  MAximum free dynamic memory (z.B. $0500)}
[WFK]    5: {Wie Prog SDIR, nur mit Korrektur in WAHL2
           und DATEIANFUEGEN, sowie
[WT4]    6:  der Proc. EXEC}
[000]    7:
[WP3]    8: {$I LIBRARY.INC}
[R64]    9: {$I WINDOW.INC}
[XU3]   10: {$I LIBRARYE.INC}
[SY8]   11: Type TSetOfChar = Set of Char;
[000]   12:
[WP3]   13: {$I READSTR.INC}
[4P3]   14: {$I TOOLS.INC}
[000]   15:
[ZOK]   16:  const    maxDateiAnzahl = 512;         {Sta
           t. Speicherstruktur}
[7A7]   17:                          ab : byte = 1;   {Für
            Filter}
[TN5]   18:                         bis : byte = 13;  {
           "     }
[TZ8]   19: Type               tstr12 = string[12];
[P15]   20:                     tstr8 = string[8];
[G65]   21:                    tdatei = record
[S53]   22:                                 name : tstr
           12;
[OX2]   23:                              passend,
[PF2]   24:                                    a : byte
           ;
[4C2]   25:                                   gr : real
           ;
[5Y3]   26:                                datum : tstr
           8;
[TUO]   27:                             end;
[4G7]   28:  var                  regs : tregs;
[FV4]   29:         Seite, MaxSeite,
[1SO]   30:                   Ind,
[366]   31:       a, s, m, st, t, mo : byte;
[A23]   32:              PassendGr,
[G15]   33:           Frei, Kap, gr : real;
[ON6]   34:          pfad, savePfad : tstr80;
[OW2]   35:                 Passend,
[YZ8]   36:   DateiAnzahl, q1, q2, j : integer;
[BP4]   37:         BSort, auswahlB,
[HD5]   38:             BF, BB, n, f : boolean;
[Z14]   39:           saveLaufwerk,
[8H5]   40:                Laufwerk : byte;
[8X2]   41:               Referenz,
[6S5]   42:                altdatei : tstr12;
[AD5]   43:                 altpfad : tstr80;
[03D]   44:                   Datei : array[1..maxDate
           iAnzahl] of tdatei;
[000]   45:
[OBK]   46: procedure chdirectory(datei : tstr80; var f
           ehler : boolean );
[Z65]   47:  var regs : tregs;
[BH1]   48:   begin
[5W4]   49:    datei := datei + #0;
[D64]   50:    regs.ah := $3b;
[A97]   51:    regs.dx := ofs(datei[1]);
[A67]   52:    regs.ds := seg(datei[1]);
[W93]   53:    msdos(regs);
[EW1]   54:    Int24F;
[NE8]   55:    fehler := ( regs.flags and 1 ) = 1;
```

```
[TUO]   56:    end;
[000]   57:
[CB5]   58: procedure wahl;
[3S5]   59:  var quit : boolean;
[000]   60:
[P44]   61:  procedure sp;
[T4D]   62:   const  st0 : string[21] = 'Gesamter Speic
          her   : ';
[E36]   63:          st1 : string[05] = ' Byte';
[XRA]   64:          st3 : string[21] = 'Belegter Speic
          her   : ';
[Y8A]   65:          st4 : string[21] = 'Freier Speiche
          r     : ';
[EOB]   66:          st2 : string[21] = 'Anzahl der Dat
          eien  : ';
[ZYA]   67:          st7 : string[19] = 'Gefilterte Dat
          eien ';
[4RA]   68:          st6 : string[21] = 'Filter-Kriteri
          um    : ';
[8C9]   69:          stA : string[21] = 'Sortieren
          : ';
[BT8]   70:          stB : string[10] = 'geschaltet';
[WG7]   71:   var      c1, c2, c3, c4 : integer;
[ZA4]   72:                       st9 : string[9];
[VW2]   73:                       stH : tstr12;
[Y53]   74:                         f : boolean;
[EG2]   75:                         y : byte;
[D92]   76:                        ch : char;
[K78]   77:                         a : array[10..19,1.
          .74] of byte;
[000]   78:
[BH1]   79:    begin
[1F2]   80:     f := false;
[KT9]   81:     speicher(c1, c2, c3, c4, LaufWerk+64);
[9K4]   82:     Frei := 1.0 * c1 * c3;
[6E3]   83:     Kap := Frei * c4;
[5P3]   84:     Frei := Frei * c2;
[E99]   85:     MakeWindow( 2, 112, 112, 32, 14, 10, 37
          , 10, a );
[X94]   86:     attribut := 112;
[8A3]   87:     SetAdr(15,11);
[PNH]   88:     SchreibeSatz(st0); MoveReal(9,0,Kap);
                  SchreibeSatz(st1);
[ZV2]   89:     NextLine;
[BSK]   90:     SchreibeSatz(st3); MoveReal(9,0,Kap-Fre
          i);     SchreibeSatz(st1);
[ZV2]   91:     NextLine;
[UZH]   92:     SchreibeSatz(st4); MoveReal(9,0,Frei);
                  SchreibeSatz(st1);
[ZV2]   93:     NextLine;
[SXC]   94:     SchreibeSatz(st2); MoveInt(9,DateiAnzah
          l);
[ZV2]   95:     NextLine;
[ONC]   96:     SchreibeSatz(st7+': '); MoveInt(9,Passe
          nd);
[ZV2]   97:     NextLine;
[CWL]   98:     SchreibeSatz(st7+'in'); MoveReal(9,0,Pa
          ssendgr); SchreibeSatz(st1);
[ZV2]   99:     NextLine;
[X9G]  100:     SchreibeSatz(st6+' '+Referenz); {Refere
          nz[0] immer 12)
[ZV2]  101:     NextLine;
[EZ3]  102:     if BSort then
[6U8]  103:      SchreibeSatz(stA+' '+'Ein'+stB)
[A11]  104:     else
[FT8]  105:      SchreibeSatz(stA+' '+'Aus'+stB);
[U78]  106:     NextLine; MemAdr := MemAdr+2;
[809]  107:     SchreibeSatz('Taste drücken');
[CL2]  108:     clear(ch);
[5L7]  109:     RestoreWindow( 14, 10, 37, 10, a );
[TUO]  110:    end;
[000]  111:
[BM8]  112:   procedure NeuesLaufwerk;
[32K]  113:    const st0 : string[33] = 'Neues Laufwerk
          (<ESC> Abbrechen) :';
[BV7]  114:    var      h : array[1..160] of byte;
[D92]  115:           ch : char;
[6D2]  116:           dr : byte;
[BH1]  117:     begin
[VT8]  118:      Move(mem[screenaddr:3360],h,160);
[SB4]  119:      attribut := 112;;
[P65]  120:      MoveStr(1,22,st0);
[NU5]  121:      gotoxy(34,22); clreol;
[G93]  122:      Cursor(1,13);
[XN3]  123:      gotoxy(35,22);
[GC8]  124:      MoveChr(35,22,chr(LaufWerk+64));
[X52]  125:      repeat
[D92]  126:       read2(ch);
[YE4]  127:       ch := upcase(ch);
[UY5]  128:       if ch = #0 then begin
[D92]  129:        read2(ch);
[9G1]  130:        ch := #32;
[TUO]  131:       end;
[HKF]  132:      until ch in [#13,#27,'A'..'Z']; {Lastd
          rive kann auch über
[OEE]  133:       Int21H Fkt 36H ermittelt werden (s. L
          IBRARY SPEICHER)}
[SL6]  134:      if ch <> #27 then begin
```

```
[CC4]  135:       attribut := 32;
[RO4]  136:       window(1,1,80,22);
[S43]  137:       gotoxy(1,1);
[ORH]  138:  (Falls DOS-Meldung "Diskette in Laufwerk..
          ." (Bei HD + 1 DD) kommt.
[M8N]  139:   Oder: Verwendung der in der LIBRARY entha
          ltenen Proc's SICHERN,ENTSICHERN)
[NM2]  140:       clrscr;
[954]  141:       if ch = #13 then
[BT3]  142:        dr := LaufWerk
[A11]  143:       else
[P93]  144:        dr := ord(ch);
[8W3]  145:       NewDrive(dr);
[PO5]  146:       Laufwerk := dr+64;
[1D2]  147:       n := true;
[GM4]  148:       BSort := false;
[1G2]  149:       BB:= false;
[YH2]  150:       BF := false;
[T76]  151:       Referenz := '????????.???';
[TUO]  152:      end;
[SK3]  153:      Cursor(32,32);
[FS8]  154:      Move(h,mem[screenaddr:3360],160);
[TUO]  155:     end;
[000]  156:
[WCG]  157:   function vergleich( Referenz, Target : ts
          tr12 ) : boolean;
[RA4]  158:    var i, j : byte;
[BH1]  159:    begin
[ZY1]  160:     i := ab;
[6A8]  161:     j := succ(length(Referenz));
[VFG]  162:     while ((Referenz[i] = Target[i]) or (R
          eferenz[i] = '?')) and
[6V2]  163:           (i < bis) do
[943]  164:      i := succ(i);
[WM5]  165:     vergleich := (i=bis);
[TUO]  166:    end;
[000]  167:
[RMB]  168:   procedure einzeln( Referenz : tstr12 );
[HL4]  169:    var  i : integer;
[4C2]  170:         gr : real;
[BH1]  171:    begin
[AD2]  172:     Seite := 1;
[VHA]  173:     if Referenz = '????????.???' then begi
          n
[H84]  174:      Passendgr := 0.0;
[OU7]  175:      Passend := DateiAnzahl;
[DX9]  176:      MaxSeite := succ(Passend div 90);
[459]  177:      for i := 1 to DateiAnzahl do begin
[2R6]  178:       datei[i].passend := 1;
[52B]  179:       Passendgr := Passendgr + datei[i].gr
          ;
[TUO]  180:      end;
[YH2]  181:      BF := false;
[DLO]  182:     end
[YS2]  183:     else begin
[VL3]  184:      Passend := 0;
[H84]  185:      PassendGr := 0.0;
[GD7]  186:      for i := 1 to DateiAnzahl do
[FHL]  187:       if boolean(Datei[i].a and 16) then
          {Passend ist bei Attr.=16 immer 1 !!!}
[G98]  188:        Passend := succ(Passend)
[A11]  189:       else
[NPE]  190:        if vergleich( Referenz, Datei[i].na
          me ) then begin
[2P6]  191:         Datei[i].passend := 1;
[BB8]  192:         Passend := succ(Passend);
[50B]  193:         Passendgr := Passendgr + Datei[i].
          gr;
[DLO]  194:        end
[A11]  195:        else
[XN6]  196:         Datei[i].passend := 0;
[DX9]  197:      MaxSeite := succ(Passend div 90);
[TUO]  198:     end;
[TUO]  199:    end;
[000]  200:
[YR5]  201:  procedure filter;
[SM9]  202:   type tsetofchar = Set of char;
[H56]  203:   var        Hilf : tstr12;
[F04]  204:              t, i : integer;
[GG2]  205:                H8 : tstr8;
[494]  206:                H3 : string[3];
[BK7]  207:                 h : array[1..3,1..60] of by
          te;
[000]  208:
[5Z8]  209:     procedure Fill( var s : tstr12 );
[PT4]  210:      var i, j : integer;
[353]  211:          Hilf : tstr12;
[BH1]  212:       begin
[G14]  213:        j := length(s);
[HP3]  214:        i := pos('*',s);
[113]  215:        if i = 0 then
[7C1]  216:         exit;
[F37]  217:        if pos(#32,s) > 0 then begin
[E59]  218:         if s[succ(i)] <> #32 then begin
[XC6]  219:          fillchar(Hilf[1],13,#32);
[5T4]  220:          Hilf := copy(s,1,i);
[LL4]  221:          Hilf[0] := chr(j);
[65D]  222:          Move(s[succ(i)],Hilf[j-pred(pos(#3
          2,s)-succ(i))],
```

```
[GS4]  223:                pos(#32,s)-succ(i));
[032]  224:             s := Hilf;
[TUO]  225:           end;
[ED1]  226:           i := 1;
[M56]  227:           while i <= j do begin
[VB6]  228:            if s[i] in [#32,'*'] then
[5C2]  229:             s[i] := '?';
[943]  230:            i := succ(i);
[TUO]  231:           end;
[DLO]  232:         end
[A11]  233:         else
[6G4]  234:          s[pos('*',s)] := '?';
[TUO]  235:       end;
[000]  236:
[BH1]  237:       begin
[X94]  238:        attribut := 112;
[X38]  239:        MakeWindow(4,112,112,32,10,10,30,3,h
       );
[OA3]  240:        SetAdr(11,10);
[6CB]  241:        SchreibeSatz('Datei-Filter'); NextLi
       ne;
[4U8]  242:        SchreibeSatz('Wildcards *,?');
[G93]  243:        Cursor(1,13);
[TC4]  244:        attribut := 25;
[X52]  245:        repeat
[9VC]  246:         readStr(Referenz,[#32,#33,#35..#42,
       #45,#46,#48..#58,#63..#90,
[SE6]  247:                                #94..#123,#125..#
       127],12,26,11,true);
[X2E]  248:         if (Referenz = '') or (Referenz[0]
       = #255) then begin
[T76]  249:          Referenz := '????????.???';
[A17]  250:          RestoreWindow(10,10,30,3,h);
[7C1]  251:          exit;
[TUO]  252:         end;
[ED1]  253:         i := 1;
[LD1]  254:         t := 0;
[VMA]  255:         while i <= length(Referenz) do begi
       n
[5V6]  256:          if Referenz[i] = '.' then
[N53]  257:           t := succ(t);
[943]  258:          i := succ(i);
[TUO]  259:         end;
[M74]  260:        until t in [0,1];
[SK3]  261:        Cursor(32,32);
[2D7]  262:        if Referenz <> '' then begin
[W86]  263:         if Referenz = '*.*' then
[826]  264:          Referenz := '????????.???'
[YS2]  265:         else begin
[GA8]  266:          t := succ(length(Referenz));
[UC6]  267:          fillchar(Hilf[1],12,#32);
[6P3]  268:          Hilf[0] := #12;
[O76]  269:          i := pos('.',Referenz);
[SG4]  270:          if i > 0 then begin
[SG4]  271:           if i > 9 then begin
[LZ6]  272:            delete(Referenz,1,i-9);
[YE1]  273:            i := 9;
[TUO]  274:           end;
[TT9]  275:           move(Referenz[1],Hilf[1],pred(i))
       ;
[WD4]  276:           if t-i > 4 then
[MP1]  277:            t := i+4;
[F09]  278:           move(Referenz[i],Hilf[9],t-i);
[DLO]  279:          end
[A11]  280:          else
[77D]  281:           Move(Referenz[1],Hilf[1],length(R
       eferenz));
[SG4]  282:          if i > 0 then begin
[PD6]  283:           {pos('.',Referenz) > 0}
[VK5]  284:           H3 := copy(Hilf,10,12);
[412]  285:           Fill(H3);
[VZ4]  286:           H8 := copy(Hilf,1,8);
[722]  287:           Fill(H8);
[AD5]  288:           Referenz := H8+'.'+H3;
[DLO]  289:          end
[YS2]  290:          else begin
[CP2]  291:           Fill(Hilf);
[M25]  292:           Referenz := Hilf;
[TUO]  293:          end;
[TUO]  294:         end;
[ND1]  295:         ab := 1;
[A57]  296:         while Referenz[ab] = '?' do
[5T3]  297:          ab := succ(ab);
[MG6]  298:         Bis := length(Referenz);
[TF7]  299:         while Referenz[Bis] = '?' do
[544]  300:          Bis := pred(Bis);
[144]  301:         Bis := succ(Bis);
[KZ6]  302:         MoveStr(26,11,Referenz);
[706]  303:         einzeln(Referenz);
[TUO]  304:        end;
[A17]  305:        RestoreWindow(10,10,30,3,h);
[TUO]  306:       end;
[000]  307:
[VU5]  308:       procedure sorter;
[000]  309:
[THD]  310:        procedure sort( v, b : integer ); {Q
       uicksort}
[PT4]  311:         var i, j : integer;
```

```
[ZU2]  312:              vg : tstr12;
[1S2]  313:               h : tdatei;
[BH1]  314:          begin
[6P7]  315:           vg := datei[(v+b) shr 1].name;
[CU1]  316:           i := v;
[NP1]  317:           j := b;
[X52]  318:           repeat
[EA7]  319:            while datei[i].name < vg do
[943]  320:             i := succ(i);
[XA7]  321:            while datei[j].name > vg do
[733]  322:             j := pred(j);
[515]  323:            if i <= j then begin
[8S3]  324:             h := datei[i];
[DA6]  325:             datei[i] := datei[j];
[034]  326:             datei[j] := h;
[943]  327:             i := succ(i);
[733]  328:             j := pred(j);
[TUO]  329:            end;
[M53]  330:           until i > j;
[B13]  331:           if j > v then
[PE2]  332:            sort(v,j);
[VO3]  333:           if i < b then
[8C2]  334:            sort(i,b);
[TUO]  335:          end;
[000]  336:
[BH1]  337:         begin
[BC5]  338:          sort(1,DateiAnzahl);
[WH2]  339:          signal;
[TUO]  340:         end;
[000]  341:
[VTG]  342:     function Search( x1, y1 : byte; Seite :
       byte) : integer;
[CC5]  343:      var h, i, j : integer;
[BH1]  344:       begin
[DKC]  345:        j := (succ)(pred(Seite) * 90) + (pre
       d(y1)*5+x1);
[KK4]  346:        if BF then begin
[ED1]  347:         i := 1;
[7D1]  348:         h := 0;
[X52]  349:         repeat
[WK7]  350:          if Datei[i].Passend = 1 then
[543]  351:           h := succ(h);
[943]  352:          i := succ(i);
[XN2]  353:         until (h=j);
[4S4]  354:         search := pred(i);
[DLO]  355:        end
[A11]  356:        else
[E03]  357:         search := j;
[TUO]  358:       end;
[000]  359:
[NFH]  360:     procedure EXEC( nm : tstr80; pfad : tst
       r80; var n : boolean );
[000]  361:
[GZL]  362:      {Folgende Lösung des 'Exec-Problems' i
       st eine sehr einfache, aber
[AZ5]  363:       sichere Methode !}
[000]  364:
[FA9]  365:      var  Param : array[1..7] of integer;
[X33]  366:             regs : tregs;
[Y93]  367:               CC : boolean;
[YOA]  368:      const  com : string[11] = 'COMMAND.COM
       ';
[AS3]  369:             line : tstr80 = '';
[PU3]  370:             name : tstr255 = '';
[000]  371:
[GA8]  372:      procedure Error( a : byte );
[YD8]  373:       var  h : array[1..6,1..40] of byte;
[D92]  374:           ch : char;
[BH1]  375:        begin
[WH2]  376:         signal;
[C38]  377:         MakeWindow(0,112,15,32, 28,12,20,5,
       h);
[X94]  378:         attribut := 112;
[BG9]  379:         MoveStr(31,13,'Fehlerbedingung');
[N94]  380:         attribut := 120;
[VX1]  381:         Adr(29,15);
[7MC]  382:         case a of {Folgende Fehler sind mög
       lich}
[19A]  383:           2: SchreibeSatz('Prog. nicht gef.
       ');
[2GA]  384:           3: SchreibeSatz('Pfad nicht gef.'
       );
[M6B]  385:           4: SchreibeSatz('Zuviele Dateien'
       );
[OEC]  386:           5: SchreibeSatz('Zugriff verweige
       rt');
[UFB]  387:           8: SchreibeSatz('Zuwenig Speicher
       ');
[TDA]  388:          10: SchreibeSatz('Umgebung falsch'
       )
[A11]  389:          else
[OSA]  390:           SchreibeSatz('Unbekannter Fehler'
       );
[TUO]  391:         end;
[T44]  392:         Warte(112,29,16,ch);
[287]  393:         RestoreWindow( 28,12,20,5,h);
[TUO]  394:        end;
[000]  395:
```

```
[2U8]  396:       function LiesKommand : tstr80;
[Z78]  397:        var h : array[1..8,1..136] of byte;
[CU2]  398:            s : tstr80;
[BH1]  399:         begin
[CH7]  400:          MakeWindow(7,112,15,32, 8,6,68,8,h)
           ;
[X94]  401:          attribut := 112;
[XS1]  402:          Adr(19,7);
[BSK]  403:          SchreibeSatz('DOS-Befehl (Aufruf de
       s Kommandoprozessors)');
[Y03]  404:          SetAdr(9,10);
[HBH]  405:          SchreibeSatz('DOS-Befehl wie gewohn
       t eingeben !  Bsp.: DIR /P');
[ZV2]  406:          NextLine;
[D6H]  407:          SchreibeSatz('Löschen Sie "/C ", so
        kommen Sie mit dem Befehl');
[ZV2]  408:          NextLine;
[B2E]  409:          SchreibeSatz('EXIT zu diesem Progra
       mm zurück '#19);
[MC4]  410:          attribut := 15;
[7K1]  411:          s := '/C ';
[ZB8]  412:          ReadStr(s,[#32..#126],65,9,9,true);
[PT6]  413:          RestoreWindow( 8,6,68,8,h);
[TD4]  414:          LiesKommand := s;
[TUO]  415:         end;
[OOO]  416:
[ZTA]  417:       procedure Tast; (Taste drücken)
[ND8]  418:        var  h : array[1..3,1..30] of byte;
[D92]  419:             ch : char;
[BH1]  420:         begin
[TE8]  421:          MakeWindow(1,112,122,32, 30,10,15,3
       ,h);
[X94]  422:          attribut := 112;
[U9A]  423:          (Warte hier nicht zu gebrauchen)
[4K8]  424:          MoveStr(31,11,'Taste drücken');
[HL2]  425:          Delay(1000);
[D87]  426:          RestoreWindow( 30,10,15,3,h);
[D92]  427:          read2(ch);
[FD4]  428:          if ch = #0 then
[D92]  429:           read2(ch);
[TUO]  430:         end;
[OOO]  431:
[BH1]  432:       begin
[BD2]  433:        name := nm;
[2UA]  434:        while pos(' ',name) > 0 do  (Formati
       eren)
[V48]  435:         delete(name,pos(' ',name),1);
[9F2]  436:        n := false;
[TH2]  437:        CC := false;
[XK7]  438:        if name = com then begin
[XH5]  439:         line := LiesKommand;
[Y82]  440:         CC := true;
[WH9]  441:         if (line[0] = #0) or (line[0] = #25
       5) then
[7C1]  442:          exit;
[TUO]  443:        end;
[1D2]  444:        n := true;
[2H6]  445:        if pfad <> '\' then begin
[2V3]  446:         pfad := pfad + '\';
[TUO]  447:        end;
[WM5]  448:        insert(pfad,name,1);
[XTB]  449:        name := name + #0;          (Mit #0
       terminieren)
[UCL]  450:        Param[1] := 0;              (Seg(Env
       ironment) Aufrufendes Prog. gleiches
[S5B]  451:                                    Environ
       ment wir rufendes Prog.)
[99C]  452:        Param[2] := ofs(line);      (Kommand
       ozeile)
[S86]  453:        Param[3] := seg(line);
[2F8]  454:        Param[4] := $5C;            (1.FCB i
       m PSP)
[CD4]  455:        Param[5] := CSeg;
[WF8]  456:        Param[6] := $6C;            (2.FCB i
       m PSP)
[UD4]  457:        Param[7] := CSeg;
[375]  458:        with regs do begin
[7Z8]  459:         AH := $4B;                 (Int21H,
        Fkt. 4BH)
[UL6]  460:         AL := 0;                   (Execute
       )
[1H4]  461:          BX := ofs(Param);
[AH4]  462:          ES := seg(Param);
[3T4]  463:          DX := ofs(name[1]);
[BW3]  464:          DS := Seg(name);
[TUO]  465:        end;
[CC4]  466:        attribut := 32;
[NF2]  467:        ClrScr;
[123]  468:        Cursor(3,8);
[W53]  469:        MsDos(regs);
[L8F]  470:        if (regs.flags and 1) = 1 then  (Car
       ry Flag gesetzt ?)
[G84]  471:         error(regs.al)
[A11]  472:        else
[OX9]  473:         if CC then (KOMMANDOPROZESSOR-Aufru
       f)
[6B1]  474:          Tast;
```

```
[6TG]  475:         Cursor(32,32); (Denn der dürfte wied
       er verändert sein)
[TUO]  476:       end;
[OOO]  477:
[6X3]  478: ($I PULLDOW1.INC)
[OOO]  479:
[RN9]  480:     procedure grz( x1,y1,seite : byte);
[1C4]  481:      var    i : integer;
[B54]  482:          str9 : string[9];
[BH1]  483:       begin
[VU6]  484:        i := search(x1,y1,seite);
[CC4]  485:        attribut := 32;
[MR7]  486:        str(Datei[i].gr:9:0,str9);
[YB5]  487:        MoveStr(8,23,str9);
[DM6]  488:        str(Datei[i].a:3,str9);
[D75]  489:        MoveStr(40,23,str9);
[FA8]  490:        MoveStr(54,23,Datei[i].datum);
[TUO]  491:       end;
[OOO]  492:
[PZ5]  493:       procedure NextPage;
[BH1]  494:         begin
[9L9]  495:          if Seite < MaxSeite then begin
[1D2]  496:           n := true;
[V82]  497:           BB := true;
[RE5]  498:           Seite := succ(Seite);
[TUO]  499:          end;
[TUO]  500:         end;
[OOO]  501:
[7Y5]  502:      procedure LastPage;
[BH1]  503:        begin
[FA7]  504:         if Seite > 1 then begin
[1D2]  505:          n := true;
[V82]  506:          BB := true;
[NE5]  507:          Seite := pred(Seite);
[TUO]  508:         end;
[TUO]  509:        end;
[OOO]  510:
[RL7]  511:      procedure go( ch : byte );
[BH1]  512:        begin
[RS2]  513:         case ch of
[TH9]  514:           80 : if y1 < yanz then    (runter
       )
[EFO]  515:                  g(1)
[A11]  516:                else
[KT2]  517:                 NextPage;
[Y47]  518:           72 : if y1 > 1 then       (hoch)
[HFO]  519:                  g(2)
[A11]  520:                else
[TS2]  521:                 LastPage;
[9B7]  522:           75 : if x1 > 1 then       (links)
[MFO]  523:                  g(3)
[A11]  524:                else
[UP3]  525:                 if y1 > 1 then
[UFO]  526:                  g(5)
[A11]  527:                 else
[TS2]  528:                  LastPage;
[UA9]  529:           77 : if x1 < xanz then    (rechts
       )
[RFO]  530:                  g(4)
[A11]  531:                else
[XE4]  532:                 if y1 < yanz then
[XFO]  533:                  g(6)
[A11]  534:                 else
[KT2]  535:                  NextPage;
[UC5]  536:           81 : NextPage; (Pgdn)
[KB5]  537:           73 : LastPage (Pgup)
[A11]  538:         else
[LE4]  539:         if ch = 59 then
[4GO]  540:          sp;
[TUO]  541:        end;
[TUO]  542:       end;
[OOO]  543:
[BH1]  544:      begin
[TW3]  545:       raus := false;
[NN2]  546:         ok := false;
[PH2]  547:         BB := false;
[9F2]  548:          n := false;
[FB7]  549:       invers1( a, x, y, anz, true );
[4U4]  550:       grz(x1,y1,seite);
[X52]  551:       repeat
[862]  552:        read2( ch );
[YM5]  553:        if ( ch = #0 ) then begin
[862]  554:         read2( ch );
[4U2]  555:         go(ord(ch));
[4U4]  556:         grz(x1,y1,seite);
[DLO]  557:       end
[YS2]  558:       else begin
[YE4]  559:        ch := upcase(ch);
[RS2]  560:        case ch of
[XZF]  561:          #13 : if boolean(Datei[search(x1
       ,y1,seite)].a and 16) then
[972]  562:                  ok := true
[A11]  563:                else
[TSC]  564:                 EXEC(Datei[search(x1,y1,s
       eite)].name,pfad,n);
[Z52]  565:          #27 : begin
[SB3]  566:                 raus := true;
[GM4]  567:                 BSort := false;
```

```
[TU0]  568:                    end;
[8A5]  569:              #78 : neuesLaufwerk;
[LE0]  570:              #52,
[TE0]  571:              #54,
[ZE0]  572:              #56,
[F64]  573:              #60 : go(ord(ch));
[M52]  574:              #70 : begin
[EL2]  575:                     Filter;
[1D2]  576:                     n := true;
[V82]  577:                     BB := true;
[392]  578:                     BF := true;
[TU0]  579:                    end;
[M08]  580:              #83 : if not BSort then begin
[TN2]  581:                     sorter;
[1D2]  582:                     n := true;
[V82]  583:                     BB := true;
[414]  584:                     BSort := true;
[DL0]  585:                    end
[YS2]  586:                    else begin
[GM4]  587:                     BSort := false;
[1D2]  588:                     n := true;
[V82]  589:                     BB := true;
[TU0]  590:                    end;
[TU0]  591:            end;
[TU0]  592:          end;
[TS6]  593:         until ok or raus or n;
[MC4]  594:         attribut := 15;
[TU0]  595:        end;
[000]  596:
[ZY5]  597:   procedure auswahl;
[KMB]  598:     const        a1 = 15;    a = 32;     x =
        2; y = 2; mxs = 74;
[617]  599:                  mys = 23; mxss = 148; mys1 =
        21;
[9P6]  600:                Leer : string[6] = 'Leer !';
[NM5]  601:    var           i : integer;
[KH4]  602:        xs, ys1, ys : byte;
[OT8]  603:                  h1 : array[1..mys1,1..mxss]
        of byte;
[0U9]  604:                   h : array[1..mys,1..mxss]
       of byte;
[000]  605:
[PB7]  606:    procedure moveDateien;
[4M7]  607:     var i, j, g, xa, ya : integer;
[BH1]  608:      begin
[4R8]  609:       if BF and (MaxSeite > 1) then begin
[ZP6]  610:        j := succ(pred(Seite) * 90);
[ED1]  611:        i := 1;
[6D1]  612:        g := 0;
[X52]  613:        repeat
[WK7]  614:         if Datei[i].Passend = 1 then
[143]  615:          g := succ(g);
[943]  616:         i := succ(i);
[XN2]  617:        until (g=j);
[333]  618:        i := pred(i);
[DL0]  619:      end
[A11]  620:       else
[YP6]  621:        i := succ(pred(Seite) * 90);
[AE1]  622:       xa := 1;
[BE1]  623:       ya := 1;
[6D1]  624:       g := 0;
[7H7]  625:       fillchar( h1,sizeof(h1),32 );
[2K6]  626:       if Passend > 0 then begin
[X52]  627:        repeat
[GF9]  628:         if Datei[i].passend = 1 then begin
[143]  629:          g := succ(g);
[K56]  630:          for j := 1 to 12 do begin
[ZTE]  631:           h1[ya,xa + pred(j) shl 1] := ord(
       Datei[i].name[j]);
[CA9]  632:           if boolean(Datei[i].a and 16) the
       n
[SZ8]  633:            h1[ya,succ(xa + pred(j) shl 1)]
       := 15
[TU0]  634:          end;
[M96]  635:          if xa = 121 then begin
[AE1]  636:           xa := 1;
[PW3]  637:           ya := succ(ya);
[DL0]  638:          end
[A11]  639:          else
[FX2]  640:           xa := xa + 30;
[TU0]  641:         end;
[943]  642:         i := succ(i);
[G19]  643:        until (i > DateiAnzahl) or (g = 90);
[ZX3]  644:        if xa = 1 then
[FN4]  645:         ys := ya   (pred)
[A11]  646:        else
[7K4]  647:         ys := succ(ya);  {}
[DL0]  648:       end
[A7L]  649:       else begin   (Keine Datei gef. (Kann
       nur im Root-Dir passieren)
[586]  650:        for j := 1 to 6 do begin
[H1C]  651:         h1[ya,xa + pred(j) shl 1] := ord(Le
       er[j]);
[SZ8]  652:         h1[ya,succ(xa + pred(j) shl 1)] :=
       15
[TU0]  653:        end;
[UX3]  654:        ys := succ(ya);
[TU0]  655:       end;
```

```
[TU0]  656:      end;
[000]  657:
[8F6]  658:    procedure dateiwahl;
[EG8]  659:     var  akt, am, m, anz : integer;
[D92]  660:                       ch : char;
[FR4]  661:                  f, rq : boolean;
[DB4]  662:         aq, xq, yq, x1q,
[M34]  663:           y1q, mq, anzq,
[GU4]  664:             xanzq, yanzq : byte;
[000]  665:
[H0C]  666:         procedure MoveLinie( att, a, b, y, c
       : byte );
[5T4]  667:          var i : integer;
[BH1]  668:          begin
[TL4]  669:           a := pred(a) shl 1;
[T85]  670:           i := (pred(y)*5) shl 5;
[VC2]  671:           b := b shl 1;
[X52]  672:           repeat
[G58]  673:            mem[ screenaddr : i+a ] := c;
[933]  674:            a := succ(a);
[UG9]  675:            mem[ screenaddr : i+a ] := att;
[933]  676:            a := succ(a);
[VC3]  677:           until a >= b;
[TU0]  678:          end;
[000]  679:
[DB5]  680:         procedure ober;
[BH1]  681:          begin
[X94]  682:           attribut := 112;
[Z15]  683:           MoveChr(x,2,'┌');
[C68]  684:           MoveLinie(112,succ(x),xs,y,196);
[1A7]  685:           MoveChr(succ(xs),2,'┐');
[TU0]  686:          end;
[000]  687:
[8N7]  688:         procedure seitenunter;
[W63]  689:          var i : byte;
[BH1]  690:          begin
[X94]  691:           attribut := 112;
[1P8]  692:           for i := 3 to succ(ys) do begin
[UZ4]  693:            MoveChr(x,i,'|');
[LB7]  694:            MoveChr(succ(xs),i,'|');
[TU0]  695:           end;
[H55]  696:           MoveChr(x,ys+2,'└');
[45C]  697:           MoveLinie(attribut,succ(x),xs,ys
       +2,196);
[D97]  698:           MoveChr(succ(xs),ys+2,'┘');
[TU0]  699:          end;
[000]  700:
[KK5]  701:       begin (dateiwahl)
[B84]  702:        attribut := a;
[YN1]  703:          akt := 1;
[TK1]  704:          y1q := 1;
[SK1]  705:          x1q := 1;
[8B3]  706:           xq := succ(x);
[DB3]  707:           yq := succ(y);
[UG1]  708:           mq := 15;
[WA2]  709:        anzq := 12;
[TK4]  710:        if Passend > 5 then
[1C2]  711:         xanzq := 5
[A11]  712:        else
[RH4]  713:         if Passend = 0 then
[TB2]  714:          xanzq := 1
[A11]  715:         else
[045]  716:          xanzq := Passend;
[MX4]  717:        yanzq := pred(ys);
[9D2]  718:        rq := true;
[X52]  719:        repeat
[D12]  720:         ys1 := ys+2;
[C9B]  721:         move(mem[screenaddr:160+pred(x) shl
       1],h,
[L99]  722:              (pred(ys)*5) shl 5+pred(x) shl
       1);
[B84]  723:         attribut := a;
[HA1]  724:         ober;
[CL4]  725:         attribut := a1;
[KZ7]  726:         MoveStr(x,pred(y),'Pfad: ');
[62E]  727:         MoveStr(x+5,pred(y),chr(Laufwerk+64)
        + ':' + pfad);
[HK4]  728:         if Passend < 5 then
[RH4]  729:          if Passend = 0 then
[FP1]  730:           anz := 24
[A11]  731:          else
[XW5]  732:           anz := pred(Passend)*30+24
[A11]  733:         else
[8Z1]  734:          anz := 144;
[VU7]  735:         am := (pred(y)*5) shl 5 + x shl 1;
[5N6]  736:         for i := 1 to ys do begin
[SU4]  737:          m := am + (i*5) shl 5;
[56A]  738:          move(h1[i],mem[screenaddr:m],anz);
[TU0]  739:         end;
[8N3]  740:         seitenUnter;
[6NE]  741:         wahl2(aq,xq,yq,x1q,y1q,8,mq,anzq,xan
       zq,yanzq,rq);
[035]  742:         if n then begin
[1H5]  743:          if not BB then begin
[U04]  744:           altDatei := '';
[S64]  745:           altPfad := '\';
[DL0]  746:          end
```

```
[A11]  747:          else
[9F2]  748:           n := false;
[KF5]  749:          auswahlB := true;
[7C1]  750:          exit;
[TUO]  751:         end;
[GK5]  752:         if not rq then begin
[955]  753:          altPfad := pfad;
[22E]  754:          altDatei := Datei[search(x1q,y1q,se
       ite)].name;
[KF5]  755:          auswahlB := true;
[7C1]  756:          exit;
[TUO]  757:         end;
[6M2]  758:        until rq;
[TUO]  759:       end;
[000]  760:
[7M5]  761:     begin {auswahl}
[RZ3]  762:       Window(1,1,80,22);
[NM2]  763:       clrscr;
[RZ3]  764:       Window(1,1,80,25);
[E55]  765:       if Passend >= 5 then
[502]  766:        xs := mxs
[A11]  767:       else
[RH4]  768:        if Passend = 0 then
[EF1]  769:         xs := 17
[A11]  770:        else
[W26]  771:         xs := pred(Passend*15);
[GL3]  772:       moveDateien;
[P33]  773:       dateiwahl;
[8EB]  774:       move(h.mem[screenaddr:160+pred(x) shl
       1],
[L99]  775:            (pred(ys)*5) shl 5+pred(x) shl 1
       );
[TUO]  776:     end;
[000]  777:
[1YN]  778:  procedure dateiSuchen( ersten : boolean; v
       ar groesse : real; var att : byte;
[H2C]  779:                          var dateiname : tst
       r80; var datum : tstr8;
[CG6]  780:                          var fehler : boolea
       n );
[36A]  781:   Type       ast = array[1..80] of byte;
[E76]  782:   Var  h1, h2, i : integer;
[N62]  783:              as : ast;
[9G2]  784:               t : byte;
[X33]  785:            regs : tregs;
[V74]  786:            hilf : string[2];
[BH1]  787:    begin
[Y36]  788:     if ersten then begin
[375]  789:      with regs do begin
[AM7]  790:       dateiname := dateiname + #0;
[XF1]  791:       Ah := $1A;
[1Y3]  792:       DS := Seg(as[1]);
[VX3]  793:       DX := Ofs(as[1]);
[W33]  794:       MSDos(regs);
[1G1]  795:       Ah := $4E;
[CE1]  796:       CX := 30;
[8B6]  797:       DS := Seg(dateiname[1]);
[2B6]  798:       DX := Ofs(dateiname[1]);
[WW2]  799:       MSDos(Regs);
[TUO]  800:      end;
[DLO]  801:     end
[YS2]  802:     else begin
[A14]  803:      regs.ah := $4F;
[W33]  804:      MSDos(regs);
[TUO]  805:     end;
[G56]  806:     fehler := regs.al <> 0;
[ER1]  807:     INT24F;
[ED7]  808:     if not fehler then begin
[F23]  809:      att := as[22];
[PZ7]  810:      Move(as[31],Dateiname[1],13);
[E05]  811:      Dateiname[0] := #13;
[KKC]  812:      dateiname[0] := chr(pred(pos(#0,datein
       ame)));
[2F4]  813:      move(as[27],h2,2);
[OF4]  814:      move(as[29],h1,2);
[CM3]  815:      if h2 < 0 then
[HF3]  816:       h1 := succ(h1);
[V57]  817:      groesse := 6.5536e4 * h1 + h2;
[FS4]  818:      t := as[25] and 31;
[TA4]  819:      str(t:2,datum);
[TY4]  820:      datum := datum+'.';
[WEA]  821:      t := (as[26] and 1) shl 3 + as[25] shr
       5;
[MM3]  822:      str(t:2,hilf);
[9Y6]  823:      datum := datum + Hilf + '.';
[TB5]  824:      t := as[26] shr 1+80;
[MT2]  825:      str(t,Hilf);
[UA6]  826:      datum := datum + Hilf;
[TUO]  827:     end;
[TUO]  828:    end;
[000]  829:
[3RN]  830:   procedure DateiAnfuegen( dat : tstr12; a
       : byte; gr : real; datum : tstr8 );
[5T4]  831:    var i : integer;
[BH1]  832:     begin
[K4L]  833:      if boolean(a and 8) or boolean(a and 4
       ) then {Volume or System}
[7C1]  834:       exit;
```

```
[XA6]  835:      if (dat <> '.') then begin
[LPC]  836:       if (pos('.COM',dat) <> 0) or (pos('.E
       XE',dat) <> 0) or
[BLB]  837:           boolean(a and 16) then begin
                  {Dir}
[VK9]  838:        DateiAnzahl := succ(DateiAnzahl);
[TUC]  839:        fillchar(Datei[DateiAnzahl].name[1],
       12,#32);
[ZT6]  840:        if dat[1] <> '.' then begin
[A54]  841:         i := pos('.',dat);
[SG4]  842:         if i > 0 then begin
[AUD]  843:          move(dat[1],Datei[DateiAnzahl].nam
       e[1],pred(i));
[62K]  844:          move(dat[i],Datei[DateiAnzahl].nam
       e[9],byte(dat[0])-pred(i));
[DLO]  845:         end
[A11]  846:         else
[6FF]  847:          move(dat[1],Datei[DateiAnzahl].nam
       e[1],byte(dat[0]));
[5C9]  848:         Datei[DateiAnzahl].name[0] := #12;
[DLO]  849:        end
[A11]  850:        else
[7WC]  851:         Datei[DateiAnzahl].name := '<Überge
       ord.>';
[T67]  852:        Datei[DateiAnzahl].a := a;
[L18]  853:        Datei[DateiAnzahl].gr := gr;
[BKA]  854:        Datei[DateiAnzahl].datum := datum;
[HY7]  855:        Passendgr := Passendgr + gr;
[TG9]  856:        Datei[DateiAnzahl].passend := 1;
[TUO]  857:       end;
[TUO]  858:      end;
[TUO]  859:     end;
[000]  860:
[KGE]  861:   procedure suchen( dat : tstr12; var f : b
       oolean );
[4U3]  862:    var     a : byte;
[4C2]  863:            gr : real;
[5Y3]  864:         datum : tstr8;
[BH1]  865:     begin
[K6B]  866:      dateisuchen(true,gr,a,dat,datum,f);
[V75]  867:      if not f then begin
[X52]  868:       repeat
[KV8]  869:        DateiAnfuegen(dat,a,gr,datum);
[SZA]  870:        dateisuchen(false,gr,a,dat,datum,f);
[RF2]  871:       until f;
[TUO]  872:      end;
[AD2]  873:      Seite := 1;
[OU7]  874:      Passend := DateiAnzahl;
[DX9]  875:      MaxSeite := succ(Passend div 90);
[EZ3]  876:      if BSort then
[TN2]  877:       sorter;
[TUO]  878:     end;
[000]  879:
[H95]  880:   procedure bild;
[6G6]  881:    var str4 : string[4];
[BH1]  882:     begin
[NM2]  883:      clrscr;
[VO4]  884:      attribut := 1;
[XMA]  885:      MoveStr(50,25,'(c) A. Baqué   Bonn, No
       v.'87');
[324]  886:      attribut := 9;
[3H7]  887:      MoveStr(32,25,'EXECUTE Ver. 1.4');
[MC4]  888:      attribut := 15;
[HA5]  889:      MoveStr(1,25,'<ESC>');
[D94]  890:      MoveChr(44,24,'N');
[ZK4]  891:      MoveStr(1,24,'F1');
[594]  892:      MoveChr(21,24,'S');
[994]  893:      MoveChr(61,24,'F');
[NP7]  894:      MoveStr(70,24,#24#25#27#26#32#17#217);
[CC4]  895:      attribut := 32;
[UOA]  896:      MoveStr(7,25,'Programm verlassen');
[U16]  897:      MoveStr(62,24,'ilter');
[3K8]  898:      MoveStr(45,24,'eues Laufwerk');
[LG8]  899:      MoveStr(4,24,'Speicher-Stat.');
[D7A]  900:      MoveStr(22,24,'ortieren nach Namen');
[836]  901:      MoveStr(1,23,'Große :');
[PF5]  902:      MoveStr(18,23,'Byte');
[ZO9]  903:      MoveStr(25,23,'Dateiattribut :');
[Y16]  904:      MoveStr(46,23,'Datum :');
[TUO]  905:     end;
[000]  906:
[A9D]  907:   procedure neu( dat : tstr12; var weg : bo
       olean );
[BH1]  908:     begin
[LX2]  909:      weg := false;
[035]  910:      if n then begin
[9F2]  911:       n := false;
[X56]  912:       aktdrive(Laufwerk);
[TUO]  913:      end;
[7D5]  914:      auswahlB := false;
[HU7]  915:      if dat = '<Übergeord.>' then
[HV1]  916:       dat := '..'
[A11]  917:      else
[SP5]  918:       while pos(' ',dat) <> 0 do
[M57]  919:        delete(dat,pos(' ',dat),1);
[VG4]  920:      if pfad <> '\' then
[WA5]  921:       pfad := pfad + '\' + dat
[A11]  922:      else
```

```
[104]  923:        pfad := pfad + dat;
[9PD]  924:       if (pfad[length(pfad)] = '\') and (pfa
         d <> '\') then
[H58]  925:        delete(pfad,length(pfad),1);
[NM2]  926:       clrscr;
[826]  927:       chdirectory(pfad,f);
[025]  928:       if f then begin
[W52]  929:        pfad := '';
[XE2]  930:        weg := true;
[TUO]  931:       end;
[F91]  932:       bild;
[CB6]  933:       if dat = '..' then begin
[EB4]  934:        aktdir( pfad, 0 );
[D55]  935:        insert('\',pfad,1)
[TUO]  936:       end;
[1M4]  937:       if pfad = '' then
[PE2]  938:        pfad := '\';
[B94]  939:       DateiAnzahl := 0;
[DR4]  940:       if not(weg) then
[AY3]  941:        suchen('*.*',f);
[XF8]  942:       if BF and not(weg) then begin
[706]  943:        einzeln(Referenz);
[TUO]  944:       end;
[TUO]  945:      end;
[000]  946:
[BH1]  947:   begin
[1H5]  948:    if not BB then begin
[H84]  949:     Passendgr := 0.0;
[VL3]  950:     Passend := 0;
[BR4]  951:     pfad := altpfad;
[SS5]  952:     neu(altdatei,quit);
[DLO]  953:    end
[YS2]  954:    else begin
[PH2]  955:     BB := false;
[7D5]  956:     auswahlB := false;
[OX3]  957:     quit := false;
[TUO]  958:    end;
[2D7]  959:    if not(quit) then begin
[892]  960:     if BF then
[706]  961:      einzeln(Referenz);
[9Y2]  962:     auswahl;
[DLO]  963:    end
[A11]  964:    else
[7D5]  965:     auswahlB := false;
[TUO]  966:   end;
[000]  967:
[AFC]  968:  begin   (CBreak:=false bringt hier nichts)
[FTA]  969:   ctrlbreak(true,ctrlbreakstatus);
[MP7]  970:   ctrlbreak(false,trueB);
[OB3]  971:   wx := wherex;
[6B3]  972:   wy := wherey;
[7C5]  973:   CursorFrage(0,Ca,Ce);
[DK2]  974:   screen;
[SK3]  975:   Cursor(32,32);
[8S8]  976:   move(mem[screenaddr:0],h,4000);
[WZ1]  977:   int24S;
[SA4]  978:   altpfad := '\';
[U44]  979:   altdatei := '';
[GM4]  980:   BSort := false;
[1G2]  981:   BB:= false;
[YH2]  982:   BF := false;
[9F2]  983:   n := false;
[T76]  984:   Referenz := '????????.???';
[LV7]  985:   aktdrive(saveLaufwerk);
[D68]  986:   Laufwerk := SaveLaufwerk;
[586]  987:   aktdir( savepfad, 0 );
[X52]  988:   repeat
[8A1]  989:    wahl;
[BT5]  990:   until not auswahlB;
[PZ1]  991:   int24R;
[B74]  992:   gotoxy(wx,wy);
[3W3]  993:   Cursor(Ca,Ce);
[OL8]  994:   move(h,mem[screenaddr:0],4000);
[XU7]  995:   Newdrive(saveLaufwerk);
[KO9]  996:   chdirectory( '\'+savepfad, f );
[99B]  997:   CtrlBreak(false,CtrlBreakStatus);
[5TO]  998:  end.
```

Listing 21. Der Quellcode für »Eraser« zum sicheren Löschen von Dateien

Datum- und Zeitänderung von Datei-Einträgen

Sie kennen sicher den Fall, daß eine Datei aktualisiert wurde, ohne daß vorher das Systemdatum richtig eingestellt war. Das Programm »Filedate« (Listing 22) hilft Ihnen beim nachträglichen Ändern des Datum- und Zeiteintrags dieser Datei. Sie können natürlich ebenfalls alle Dateien einer Diskette auf das gleiche Datum und die gleiche Zeit setzen. Die Prozedur besitzt folgende Syntax:

```
Filedate [Laufw.:][Pfad1]Dateiname.Erw. [/P]
```

Der Dateiname darf ebenfalls Joker und Wildcards enthalten. Fehlt die Angabe »Pfad1«, so wird wiederum der aktuelle Pfad angenommen. Auch dieses Programm ist gegen Datenverlust abgesichert und läßt sich mit [Ctrl-C] abbrechen. Die Angabe des Parameters [/P] bewirkt vor jeder Änderung die Frage an den Benutzer, ob die angegebene Änderung wirklich durchgeführt werden soll.

(Achim Baqué/sp)

```
[B55]    0: {$R-,V-,I-,K-,D-,C-,U-}
[000]    1:
[DUE]    2: {FILEDATE 18.11.87, (Geändert für DOS inter
           national 8.1.88)}
[000]    3:
[WP3]    4: {$I LIBRARY.INC}
[R64]    5: {$I WINDOW.INC}
[XU3]    6: {$I LIBRARYE.INC}
[4P3]    7: {$I TOOLS.INC}
[WRD]    8: {$I READDATEN.INC} {Datum/Zeit Einleserouti
           ne}
[000]    9:
[TT5]   10: procedure reading;
[VC8]   11:  type                  tstr80 = string[80];
[215]   12:  var           BFrage, BVerz,
[P55]   13:                      fehl, io : boolean;
[FZ3]   14:                         h, d : integer;
[A64]   15:                   drive, y, a : byte;
[C84]   16:                datei1, datei4,
[3A5]   17:                  datei5, pfad : tstr80;
[D92]   18:                            ch : char;
[PA4]   19:                            s4 : string[4];
[WF5]   20:                     saveregs : tregs;
[TD6]   21:                            as : array[1..43]
          of byte;
[000]   22:
[WHP]   23:  procedure dateiSuchen( ersten : boolean; v
          ar a : byte; var dateiname : tstr80;
[886]   24:                              var fehler : boolea
          n);
[Z65]   25:    var regs : tregs;
[BH1]   26:    begin
[Y36]   27:     if ersten then begin
[375]   28:      with regs do begin
[AM7]   29:       dateiname := dateiname + #0;
[XF1]   30:       Ah := $1A;
[1Y3]   31:       DS := Seg(as[1]);
[VX3]   32:       DX := Ofs(as[1]);
[W33]   33:       MSDos(regs);
[1G1]   34:       Ah := $4E;
[CE1]   35:       CX := 30;
[8B6]   36:       DS := Seg(dateiname[1]);
[2B6]   37:       DX := Ofs(dateiname[1]);
[WW2]   38:       MSDos(Regs);
[DA4]   39:       fehler := al <> 0;
[TUO]   40:      end;
[DLO]   41:     end
[YS2]   42:     else begin
[A14]   43:      regs.ah := $4F;
[W33]   44:      MSDos(regs);
[G56]   45:      fehler := regs.al <> 0;
[TUO]   46:     end;
[ED7]   47:     if not Fehler then begin
[7R2]   48:      a := as[22];
[E15]   49:      dateiname[0] := #13;
[P58]   50:      move(as[31],dateiname[1],13);
[G3D]   51:      dateiname[0] := chr( pred(pos(#0,datei
          name)) );
[TUO]   52:     end;
[TUO]   53:    end;
[000]   54:
[SEM]   55:  procedure dateiDZ( dateiname : tstr80; a :
          byte; var fehler : boolean );
[GV8]   56:   var zugriffswort : integer;
[X33]   57:               regs : tregs;
[N5F]   58:    begin  (Datei in ihrem Datum und ihrer Z
          eit ändern)
[AM7]   59:     dateiname := dateiname + #0;
[375]   60:     with regs do begin
[9Y1]   61:      ax := $3d01;
[8N6]   62:      ds := seg(dateiname[1]);
[2N6]   63:      dx := ofs(dateiname[1]);
[W93]   64:      msdos(regs);
[FE7]   65:      fehler := ( flags and 1 ) = 1;
[OE5]   66:      zugriffswort := ax;
[TUO]   67:     end;
[ED7]   68:     if not fehler then begin
[LX4]   69:      regs := saveregs;
[EF8]   70:      regs.bx := zugriffswort;
[W93]   71:      msdos(regs);
[NE8]   72:      fehler := ( regs.flags and 1 ) = 1;
[TUO]   73:     end;
[4XE]   74:     with regs do begin   {Auf jeden Fall pr
          obieren !!!}
[7Y1]   75:      ax := $3e00;
[VF5]   76:      bx := zugriffswort;
```

```
[W93]   77:       msdos(regs);
[FE7]   78:       fehler := ( flags and 1 ) = 1;
[TUO]   79:      end;
[TUO]   80:     end;
[000]   81:
[HEB]   82:   procedure init( d : tdatum; z : tzeit );
[5T4]   83:    var i : integer;
[BH1]   84:     begin
[A6P]   85:      with saveregs do begin (Wird für alle z
        u ändernden Dateien gebraucht)
[WW1]   86:       ax := $5701;
[814]   87:       dx := d.jahr - 1980;
[AP3]   88:       dx := dx shl 9;
[8H4]   89:        i := d.monat shl 5;
[9MD]   90:       dx := dx + i + d.tag;       (Datums/Zei
        tformat in DOS)
[3K4]   91:       cx := z.stunde;
[CK3]   92:       cx := cx shl 11;
[KW4]   93:        i := z.minute shl 5;
[006]   94:       cx := cx + i + z.sekunde;
[TUO]   95:      end;
[TUO]   96:     end;
[000]   97:
[VEA]   98:   procedure scroll( var y : byte );
[BH1]   99:     begin
[CB4]  100:      if y >= 25 then
[8H2]  101:       rollen1
[A11]  102:      else
[963]  103:       y := succ(y);
[893]  104:      gotoxy(1,y);
[TUO]  105:     end;
[000]  106:
[WEG]  107:   procedure auswert( var str255 : tstr255; v
        ar f : boolean );
[W63]  108:    var i : byte;
[BH1]  109:     begin
[X52]  110:      repeat
[LZ4]  111:       i := pos('/',str255);
[1F2]  112:       f := false;
[SG4]  113:       if i > 0 then begin
[H7A]  114:        case upcase(str255[succ(i)]) of
[9L4]  115:         #80 : BFrage := true;
[A11]  116:         else
[TC2]  117:          f := true;
[TUO]  118:        end;
[7N4]  119:        delete(str255,i,2);
[TUO]  120:       end;
[503]  121:      until i = 0;
[TUO]  122:     end;
[000]  123:
[6S5]  124:   procedure infos;
[YBC]  125:    const s1 : string[14] = '[Laufw.:][Pfad';
[8V9]  126:          s2 : string[17] = '1]Dateiname.Erw.
       ';
[BH1]  127:     begin
[6N2]  128:      scroll(y);
[MC4]  129:      attribut := 15;
[K37]  130:      MoveStr(1,y,s1+']FILEDATE');
[T14]  131:      attribut := 7;
[EX6]  132:      MoveStr(25,y,s1+s2+'[/P]');
[6N2]  133:      scroll(y);
[6N2]  134:      scroll(y);
[VCG]  135:      MoveStr(1,y,s1+s2+': Dateien, deren Dat
       um/Zeit geändert');
[6N2]  136:      scroll(y);
[H6K]  137:      MoveStr(34,y,'werden sollen. Darf Wildc
       ards enthalten (*,?).');
[MY5]  138:      scroll(y);scroll(y);
[UNL]  139:      MoveStr(1,y,'Fehlt Pfad1, so wird der a
       ktuelle Pfad angenommen !');
[MY5]  140:      scroll(y);scroll(y);
[2GT]  141:      MoveStr(1,y,'/P Der Benutzer wird bei j
       eder Datei gefragt, ob geändert werden soll
        !');
[MY5]  142:      scroll(y);scroll(y);
[O2M]  143:      MoveStr(1,y,'Das Programm kann mit <Ctr
       l,C> abgebrochen werden.');
[6N2]  144:      scroll(y);
[TUO]  145:     end;
[000]  146:
[AL5]  147:   procedure Falsch;
[BH1]  148:     begin
[AGD]  149:      MoveStr(1,y,'Parameter oder -anzahl fal
       sch !');
[PZ1]  150:      int24R;
[99B]  151:      CtrlBreak(false,CtrlBreakStatus);
[5B1]  152:      halt;
[TUO]  153:     end;
[000]  154:
[RBA]  155:   procedure upper( var s : tstr255 );
[W63]  156:    var i : byte;
[BH1]  157:     begin
[HR6]  158:      for i := 1 to byte(s[0]) do
[5A5]  159:       s[i] := upcase(s[i]);
[TUO]  160:     end;
[000]  161:
[3L8]  162:   procedure DZ( var r : boolean );
[KC5]  163:    var datum : tdatum;
[153]  164:        zeit : tzeit;
```

```
[BH1]  165:     begin
[6N2]  166:      scroll(y);
[MC4]  167:      attribut := 15;
[5D2]  168:      r := true;
[FUB]  169:      ReadDaten(1,y,datum,zeit,false,true,r);
[VF2]  170:      if r then
[7C1]  171:       exit;
[6N2]  172:      scroll(y);
[5D2]  173:      r := true;
[GEB]  174:      ReadDaten(1,y,datum,zeit,true,true,r);
[VF2]  175:      if r then
[7C1]  176:       exit;
[6N2]  177:      scroll(y);
[8D5]  178:      init(datum,zeit);
[TUO]  179:     end;
[000]  180:
[EP4]  181:   procedure msg;
[BH1]  182:     begin
[6N2]  183:      scroll(y);
[FK2]  184:      str(d:4,s4);
[A8M]  185:      MoveStr(1,y,s4+' Datei(en) in ihrem Datu
       m und ihrer Zeit geändert !');
[6N2]  186:      scroll(y);
[TUO]  187:     end;
[000]  188:
[BH1]  189:     begin
[T14]  190:      attribut := 7;
[DK2]  191:      screen;
[3D1]  192:      d := 0;
[G13]  193:      y := wherey;
[DX4]  194:      BFrage := false;
[DR6]  195:      if ParamCount = 0 then
[KL1]  196:       infos
[YS2]  197:      else begin
[20A]  198:       fillchar(datei1,sizeof(datei1),#32);
[PR6]  199:       if ParamCount > 2 then
[R32]  200:        Falsch
[A11]  201:       else
[NZ7]  202:        if ParamCount = 2 then begin
[EV5]  203:         datei4 := ParamStr(2);
[KK5]  204:         auswert(datei4,io);
[HM2]  205:         if io then
[MG2]  206:          Falsch;
[TUO]  207:        end;
[SU5]  208:       datei1 := ParamStr(1);
[BW3]  209:       upper(datei1);
[W52]  210:       pfad := '';
[VE2]  211:       drive := 0;
[7ZB]  212:       if (datei1[2] = ':') and (datei1[0] > #
       0) then
[BL6]  213:        drive := ord(datei1[1])-64
[A11]  214:       else
[2E5]  215:        aktdrive(drive);
[EBB]  216:       if (datei1[0] = #0) or ((datei1[1] <> '
       \') and
[818]  217:           (datei1[3] <> '\')) then begin
[ZE5]  218:        aktdir(pfad,drive);
[TF4]  219:        if pfad <> '' then
[YA5]  220:         insert('\',pfad,1);
[R09]  221:        if pfad[byte(pfad[0])] <> '\' then
[2V3]  222:         pfad := pfad + '\';
[876]  223:        insert(pfad,datei1,1);
[TUO]  224:       end;
[M75]  225:       h := length(datei1);
[4T4]  226:       datei4 := datei1;
[X52]  227:       repeat
[OT6]  228:        if datei4[h] <> '\' then
[MZ4]  229:         delete(datei4,h,1);
[Z23]  230:        h := pred(h);
[OH9]  231:       until (datei4[succ(h)] = '\') or (h = 0
       );
[ON2]  232:       DZ(fehl);
[GY6]  233:       if not fehl then begin
[1R9]  234:        dateiSuchen( true, a, datei1, fehl );
[VD7]  235:        while not fehl do begin
[3X7]  236:         if keypressed then begin
[D92]  237:          read2(ch);
[FD4]  238:          if ch = #0 then
[R52]  239:           read2(ch)
[A11]  240:          else
[OZ5]  241:          if ch = #3 then begin
[MVO]  242:           msg;
[WH2]  243:           signal;
[PZ1]  244:           int24R;
[99B]  245:           CtrlBreak(false,CtrlBreakStatus);
[5B1]  246:           halt;
[TUO]  247:          end;
[TUO]  248:         end;
[1DC]  249:         if boolean(a and 16) or boolean(a and
       8) or
[96E]  250:            boolean(a and 4) or boolean(a and
       2) then begin
[DLO]  251:         end
[YS2]  252:         else begin
[V47]  253:          datei5 := datei4+datei1;
[6N2]  254:          scroll(y);
[UY5]  255:          MoveStr(1,y,datei5);
[2M5]  256:          if BFrage then begin
[1C6]  257:           h := byte(datei5[0])+2;
```

```
[ZT7]  258:          MoveStr(h,y,'ändern [J/N]');
[HT1]  259:          h := h+14;
[893]  260:          gotoxy(h,y);
[X52]  261:          repeat
[D92]  262:           read2(ch);
[YE4]  263:           ch := upcase(ch);
[FD4]  264:           if ch = #0 then
[D92]  265:            read2(ch);
[NH5]  266:          until ch in [#3,#74,#78];
[OZ5]  267:          if ch = #3 then begin
[MVO]  268:           msg;
[WH2]  269:           signal;
[PZ1]  270:           int24R;
[99B]  271:           CtrlBreak(false,CtrlBreakStatus);
[5B1]  272:           halt;
[TUO]  273:          end;
[RU4]  274:          MoveChr(h,y,ch);
[S54]  275:          if ch = #74 then
[VU6]  276:           dateiDZ( datei5, a, io )
[A11]  277:          else
[WC2]  278:           io := true;
[DLO]  279:         end
[A11]  280:         else
[DC6]  281:          dateiDZ(datei5, a, io );
[8A4]  282:         if not io then
[N33]  283:          d := succ(d);
[TUO]  284:        end;
[YH9]  285:        dateiSuchen( false, a, datei1, fehl )
                 ;
[TUO]  286:       end;
[TUO]  287:      end;
[MVO]  288:      msg;
[TUO]  289:     end;
[TUO]  290:    end;
[OOO]  291:
[T7C]  292:  begin  {CBreak:=false bringt hier nichts}
[FTA]  293:   ctrlbreak(true,ctrlbreakstatus);
[MP7]  294:   ctrlbreak(false,trueB);
[WZ1]  295:   int24S;
[1V2]  296:   reading;
[PZ1]  297:   int24R;
[99B]  298:   CtrlBreak(false,CtrlBreakStatus);
[5TO]  299:  end.
```

Listing 22. Mit dem Programm »Filedate« ändern Sie nachträglich die Datums- und Zeiteinträge Ihrer Dateien.

Tips zur Eingabe der Dateien

Wenn Sie eines der Programme abtippen, so müssen Sie zunächst kontrollieren, welche Include-Dateien ({$I ....}) das betreffende Programm benötigt. Versuchen Sie, das Programm ohne die benötigten Include-Dateien zu compilieren, so ernten Sie lediglich eine Fehlermeldung. Einige Programmteile wiederholen sich in verschiedenen Programmen. Sie können somit einige Programmteile ohne größere Änderungen übernehmen. Das Programm »Execute« enthält zum Beispiel große Teile der Directory-Routine »Sdir«.

Copy II- Optionen Board V4.5 DM 219,–

PC Tools Deluxe DM 149,–

PC Tools V3.23 DM 79,–

Utah Fortran V1.0 DM 99,–

Utah Cobol V4.0 DM 99,–

Norton Utilities V4.0 adva. DM 249,–

Kostenlose Prospekte gibt's bei...

CWTG

CWTG Joachim Tiede
Bergstr. 13
7109 Roigheim
Tel./BTX 06298/3098
v. 17 – 19 Uhr

Profi-Tools für QuickBASIC und Turbo BASIC

Schreiben Sie schnellere, leistungsfähigere und professionellere Programme! Wir helfen Ihnen dabei mit nützlichen Tools.
Zum Beispiel:

- **Toolboxen (Fenstertechnik, Menüs, DOS-Funktionen etc.)**
- **Relationale Datenbank mit komfortablem Masken-Editor**
- **Grafik-Paket (Geschäftsgrafik und Zeichensatz-Generator)**
- **Maus-Unterstützung für Ihre Programme**

Alle Pakete mit ausführlich dokumentierten Quelltexten und Programmbeispielen. Wo erforderlich, kommen schnelle Assembler-Routinen zum Einsatz. Wollen Sie mehr aus Ihrem BASIC-Compiler herausholen? Wir informieren Sie gerne kostenlos!

Ingenieur-Büro Harald Zoschke
Berliner Str.3,
D-2306 Schönberg/Holstein
Telefon (0 43 44) 61 66

Eingetr. Warenzeichen: QuickBASIC: Microsoft: Turbo BASIC: Borland

Eine Bitte an unsere Abonnenten

Vermerken Sie bei Schriftverkehr und Zahlungen neben der vollständigen Anschrift stets Ihre Abo-Nummer.

Sie vermeiden damit unnötige Verzögerungen bei der Bearbeitung Ihres Abonnements.

Vielen Dank

Ihre DMV-Versandabteilung

Fonteditor

– Zeichensatz für 24-Nadel-Drucker

Einen eigenen Zeichensatz für seinen Drucker zu schreiben, erfordert eine Menge Rechenarbeit – außer man arbeitet mit dem Fonteditor. Dieser bietet nützliche Hilfen bei der Programmierung des frei definierbaren Zeichensatzes eines 24-Nadel-Druckers.

Haben Sie sich als stolzer Besitzer eines 24-Nadel-Druckers nicht schon immer gewünscht, das Schriftbild Ihres Druckers nach individuellen Vorstellungen zu erweitern? Wenn ja, dann hilft Ihnen von nun an der Fonteditor bei dieser Arbeit. Oft liegt nämlich eine bedeutsame Fähigkeit des Druckers brach, nämlich die Fähigkeit, eigene Zeichen und komplette Zeichensätze mit dem Computer zu entwerfen und sie anschließend dem Drucker zu übergeben.
Die meisten Druckerhandbücher beschreiben diesen sogenannten Download-Vorgang nur sehr knapp. Vielen Anwendern erscheint die Funktion umständlich, und sie verzichten deshalb darauf. Hier hilft nun der Fonteditor weiter, ein Programm (Listing), mit dessen Hilfe sich einzelne Zeichen und ganze Schriftarten entwerfen, ändern, ausprobieren und verwalten lassen. Der Fonteditor wurde für die Druckerfamilie NEC P5/P6/P7 entwickelt, läßt sich aber problemlos auf allen 24-Nadel-Druckern einsetzen, die Steuercode-kompatibel zum Epson LQ 1500 sind. Das Programm ist in Turbo Pascal geschrieben, so daß einer Erweiterung oder Veränderung nichts im Wege steht.
Um nun einen eigenen Zeichensatz zu definieren, müssen Sie sich als erstes bewußt machen, wie ein Zeichensatz druckerintern aufgebaut ist. Bei Matrixdruckern werden die Zeichen durch einzelne Punkte dargestellt. Solch eine Matrix (Bild 1) können Sie mit bestimmten Punkten ausfüllen und damit jedes beliebige Zeichen definieren. Die maximale Höhe eines Zeichens beträgt 24 Punkte, die Breite richtet sich nach der gewählten Schriftart. Die Tabelle gibt Auskunft über die in Frage kommenden Auflösungsstufen.
Um nun ein Zeichen selbst zu definieren, wählen Sie als erstes die Schriftart Draft (10 cpi und 12 cpi). Dabei ist ein Buchstabe neun Spalten breit. Wollen Sie nun beispielsweise das »M« neu definieren, so tragen Sie die einzelnen Punkte, aus denen der neue Buchstabe bestehen soll, in eine 9x24 Matrix ein (vergleiche Bild 1). Das Punktmuster müssen Sie nur noch in eine dem Drucker verständliche Befehlsfolge übersetzen.
Die 24 Punkte einer Spalte werden in drei Byte abgelegt. Jedem Bit entspricht ein Punkt. Um nun die Werte der

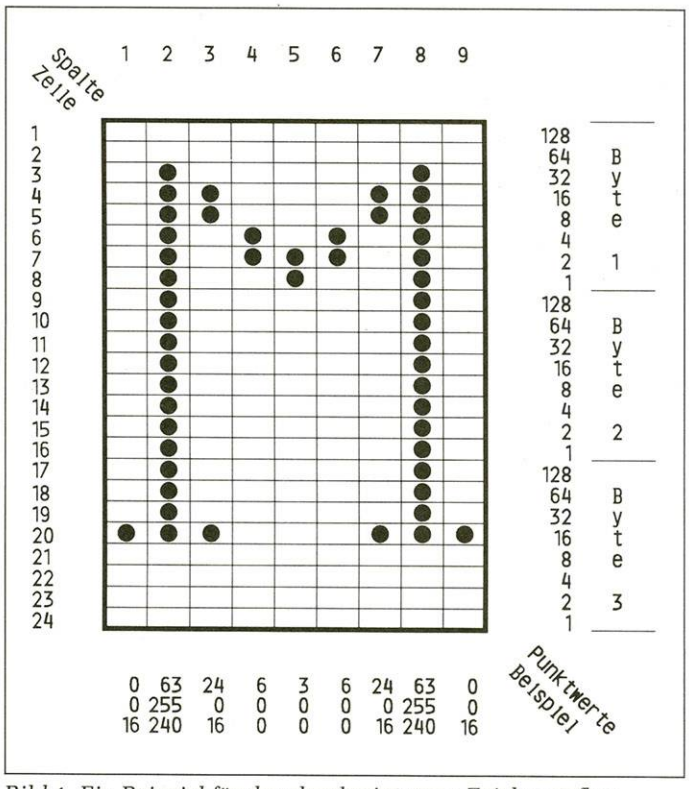


Bild 1. Ein Beispiel für den druckerinternen Zeichenaufbau

drei Byte einer jeden Spalte zu berechnen, zählen Sie – von oben beginnend – die Bits, die für einen Punkt gesetzt sind, zusammen (siehe Bild 1). Dem obersten Bit ist immer der Wert 128 zugeordnet, dem zweiten der Wert 64 und so weiter. Prinzipiell läßt sich so der Code eines Zeichens leicht bestimmen, doch bei insgesamt 27 Byte (drei Byte in jeder der neun Spalten) ist dies eine mühsame Rechnerei. Bei der Auswertung einer 37x24-Matrix (beispielsweise für LQ PS) mit 111 Byte artet dieser Vorgang regelrecht in Arbeit aus. Für solch eine stumpfsinnige Rechnerei hat Ihr Computer die besseren Voraussetzungen. Der Fonteditor übernimmt unter anderem gerade diese Additionen.
Neben den 27 Byte, die der Drucker für das »M« benötigt, sind allerdings noch weitere Angaben erforderlich (Tabelle):

- linker Rand
- Breite des Zeichens
- rechter Rand

Die Summe aus diesen drei Angaben muß immer einen bestimmten Betrag ergeben – im Draft-Modus zwölf (alle Angaben in Punktspalten). Dadurch ist gewährleistet, daß die Zeichen einen bestimmten Abstand voneinander beibehalten.

Schriftart	Maximale Matrixgröße		Linker Rand + Zeichenbreite + rechter Rand
	horizontal	vertikal	
Draft	9	24	12
Letter Quality 10	29	24	36
Letter Quality 12	25	24	30
Letter Quality PS	37	24	abhängig von tatsächlicher Zeichenbreite und gewähltem Rand

Tabelle. Auflösungsstufen bei den 24-Nadel-Druckern P5/P6/P7

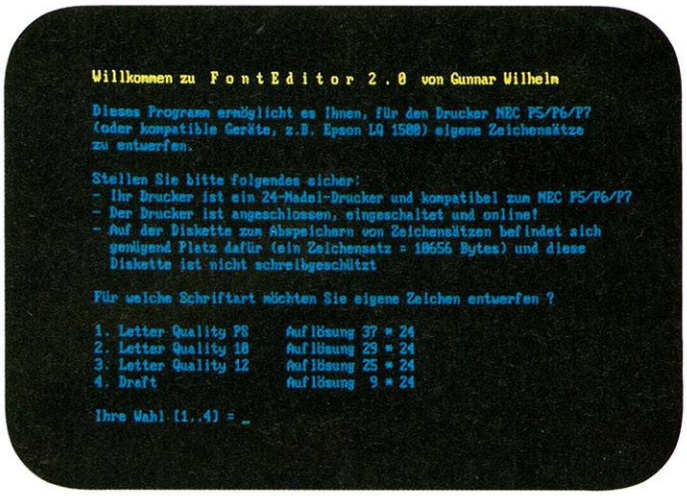


Bild 2. Das Startbild vom Fonteditor

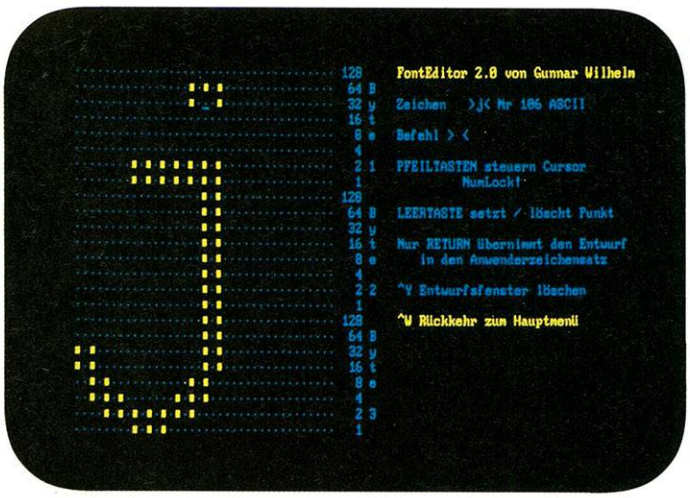


Bild 3. Das Hauptmenü des Fonteditors

Nach der ganzen Rechnerei können Sie nun das Zeichen an den Drucker senden. Die Befehlsfolge dafür lautet in Turbo Pascal

```
write (lst, chr(27), '&', chr(0), 'MM')
```

Der Befehl sagt dem Drucker, daß die nachfolgenden Bytes das Zeichen »M« ersetzen sollen.
Die Bytes sind in folgender Reihenfolge angeordnet:

1) der linke Rand (beispielsweise 2)
2) Zeichenbreite (9)
3) rechter Rand (damit die Summe 12 wird muß hier eine »1« stehen)

Danach folgen die Werte für das eigentliche Zeichen:
4) Byte 1, Byte 2 und Byte 3 aus der ersten Spalte (für das »M« 0, 0 und 16)
5) Byte 1, Byte 2, Byte 3 aus der zweiten Spalte (63, 255 und 240)
und so weiter bis
12) Byte 1, Byte 2, Byte 3 aus der neunten Spalte (0, 0 und 16)
Die entsprechende Anweisung in Turbo Pascal lautet damit

```
write (lst, chr(2), chr(9), chr(1), chr(0), chr(0), chr(16),
..., chr(0), chr(0), chr(16))
```

Das Senden dieser 30 Byte übernimmt natürlich auch der Fonteditor. Nach Aufruf des Programms sehen Sie eine Startmeldung (Bild 2). Diese Informationen sind sehr wichtig, damit das Programm korrekt arbeitet.
Schon beim Start entscheiden Sie sich für eine Schriftart, in der Sie Ihre Zeichen definieren wollen. Im Download-Druckerspeicher kann nämlich nur eine Schriftart (also Draft, LQ 10, LQ 12 oder LQ PS) stehen. Mit [1] wählen Sie beispielsweise LQ PS aus.
Als nächstes erscheint das Hauptmenü (Bild 3). Mit den Tasten [+] und [-] kennzeichnen Sie das Zeichen, das bearbeitet werden soll. Es ist in dem Eingabefenster (links in Bild 3) zu sehen. Die Breite hängt von der Schriftart ab. Das aktuelle Zeichen können Sie nun entweder bearbeiten oder an den Drucker senden (siehe unten bei Option »D«). Das Zeichen selbst wird auf dem Bildschirm oben rechts in normaler Größe zusammen mit seinem ASCII-Wert angezeigt.
[E] startet den eigentlichen Fonteditor. Das Zeichen wird, soweit es schon definiert ist, im Eingabefenster dargestellt. Mit den Cursortasten (eigentlich den Zifferntasten [2],[4],[6] und [8], da NumLock automatisch eingeschaltet wurde) steuern Sie den Cursor innerhalb des Eingabefelds. Die Leertaste setzt – beziehungsweise löscht –

Bild 4. Kaum noch Unterschiede zu einem Typenraddrucker erkennen Sie bei dem selbstdefinierten Zeichensatz »Letter«

einen Punkt an der Cursorposition, [Ctrl-Y] löscht das komplette Eingabefeld. Mit [Return] wird das Eingabefeld gelesen und das Zeichen an den Drucker geschickt. [Ctrl-W] beendet den Editor, und das Hauptmenü ist wieder aktiviert. Wollen Sie einen Zeichensatz laden, so drücken Sie [L] und geben den entsprechenden Namen ein. [S] speichert den gerade bearbeiteten Zeichensatz.

Mit [D] wird das aktuelle Zeichen an den Drucker gesandt. Haben Sie das Zeichen schon aus dem Editor heraus an den Drucker übertragen (siehe unter [E]), so müssen Sie diesen Vorgang natürlich nicht noch einmal wiederholen. Die mit [D] aufgerufene Routine ist dazu gedacht, einzelne Zeichen aus einem geladenen Zeichensatz, die nicht weiter bearbeitet werden sollen, an den Drucker zu senden.

[A] übergibt sämtliche Zeichen zwischen ASCII 33 und 127 auf einmal an den Drucker. Damit stellt [A] die D-Funktion für den kompletten Zeichensatz dar.

Mit [P] können Sie sich die Zeichen ASCII 32 bis 127 probeweise anschauen. Zeichenentwürfe lassen sich damit »vor Ort« überprüfen.

Zweimal [Ctrl-Q] beendet das Programm. Dabei gehen nicht gespeicherte Zeichensätze unwiderruflich verloren.

Bild 4 zeigt den Zeichensatz »Letter«. Diesen Zeichensatz erhalten Sie gemeinsam mit dem Programm auf der Datebox-Diskette geliefert. Entgegen der Darstellung im »P6/P7 Series Technical Reference Guide« lassen sich allerdings nicht 128, sondern nur 96 Zeichen (ASCII 32 bis 127) umdefinieren. Die Steuercodes (ASCII kleiner als 32) können natürlich nicht überschrieben werden. Desweiteren dürfen laut NEC in keiner Zeile zwei Punkte direkt nebeneinander stehen. Diesen Umstand sollten Sie bei der Gestaltung eigener Zeichensätze beachten, wenn auch diese Anweisung nicht immer zu Problemen führt.

Aus der Tabelle sehen Sie, daß die Schriftart LQ PS keine feste Zeichenbreite besitzt. Es sind maximal 37 Spalten erlaubt, aber jedes Zeichen kann beliebig darunter liegen.

Damit ist LQ PS eine Proportionalschrift, die jedem Zeichen eine eigene Breite zuteilt. Das Auffinden der tatsächlichen Zeichenbreite übernimmt wiederum der Fonteditor, so daß es gleichgültig ist, in welcher Spalte Sie mit Ihrem Zeichen beginnen und wo Sie es enden lassen. Das Programm übergibt die Zeichen immer »passend« an den Drucker.

Entsprechend gilt für alle anderen, »normalen« Schriftarten, daß die Zeichen – auch hier unabhängig von Anfangs- und Endposition – mittig justiert weitergegeben werden.

(Gunnar Wilhelm/hg)

```
[H51]    0: {$C-}
[000]    1:
[A46]    2: program FontEditor;
[000]    3:
[TUA]    4: { Zeichensatz Utility für NEC P5/P6/P7 }
[AB8]    5: {        Autor: Gunnar Wilhelm          }
[7U8]    6: {        Für DOS International          }
[Y36]    7: {              August 1987              }
[07A]    8: {    Überarbeitet im Dezember 1987      }
[000]    9:
[VM1]   10:   const
[7M1]   11:     esc = #27;
[OE1]   12:     fs  = #28;
[HG1]   13:     bel = #7;
[B85]   14:     abstand_links = 4;
[SNB]   15:     { linker Rand eines Zeichens bei LQ PS
          }
[RM5]   16:     abstand_rechts = 4;
[VY6]   17:     { dito rechts
          }
[M68]   18:     Punkt_gesetzt = #254;  {  ■  }
[Z67]   19:     Punkt_leer    = #250;  {  ·  }
[000]   20:
[A21]   21:   type
[XC4]   22:     txt = string [80];
[000]   23:
[EMO]   24:   var
[G9C]   25:     zeichen : array [32..127, 1..37, 1..3]
          of byte;
[Y1A]   26:     fenster : array [1..37, 1..24] of 0..1;
[ND4]   27:     aktuell : 32..127;
[P2B]   28:     { ASCII Nr des ausgewählten Zeichen }
[2L9]   29:     cy : 1..24;              { Cursorpositio
          n  }
[MK8]   30:     cx : 1..37;              {  im Editierfe
          ld }
[DTF]   31:     breite : 1..37;          { Breite gemäß
          gewähltem Font}
[S7F]   32:     zbreite : 1..37;         { tatsächliche
          Zeichenbreite }
[582]   33:     rand : 3..7;
[6SC]   34:     { s. NEC Technical Reference Guide 8-4:
          A+B+C - B }
[WD2]   35:     ch :char;
[HB5]   36:     dateiname : txt;
[3G7]   37:     zeichensatz : file of byte;
[B16]   38:     vorschub : boolean;
```

```
[000]   39:
[BSA]   40: {------------------\writexy----------------
          -------------}
[000]   41:
[TNC]   42:   procedure writexy (x,y : integer; s : txt
          );
[000]   43:
[BH1]   44:   begin
[HL3]   45:     gotoxy (x,y);
[P62]   46:     write (s);
[TU0]   47:   end;
[000]   48:
[E8B]   49: {------------------\clreol_xy--------------
          -------------}
[000]   50:
[W6B]   51:   procedure clreol_xy (x, y : integer);
[000]   52:
[BH1]   53:   begin
[HL3]   54:     gotoxy (x,y);
[NL2]   55:     clreol;
[TU0]   56:   end;
[000]   57:
[KHA]   58: {------------------\eingabe----------------
          -------------}
[5EL]   59: { Komfortable und universell einsetzbare Ei
          ngabeprozedur }
[000]   60:
[X8F]   61:   procedure eingabe (laenge : integer; var
          s : txt);
[000]   62:
[EMO]   63:     var
[D92]   64:       ch : char;
[MA2]   65:       wert : txt;
[1A4]   66:       s1 : string[1];
[4B4]   67:       lauf : integer;
[000]   68:
[BH1]   69:   begin
[K32]   70:     lauf := 0;
[WD1]   71:     s1 := ' ';
[M92]   72:     wert := '';
[X52]   73:     repeat
[FX3]   74:       read (kbd, ch);
[K66]   75:       if (ord (ch) in [32..127])
[B11]   76:       then
[BH1]   77:         begin
[MA5]   78:           if lauf < laenge
[B11]   79:           then
```

```
[BH1]   80:              begin
[2D2]   81:                write (ch);
[EN3]   82:                lauf := lauf + 1;
[HS2]   83:                s1 [1] := ch;
[9V6]   84:                wert := concat (wert, s1);
[TH3]   85:                if ord (ch) = 13
[B11]   86:                then
[K78]   87:                  delete (wert, length (wert)
        , 1);
[DLO]   88:              end
[A11]   89:            else
[VP2]   90:              write (bel);
[TUO]   91:          end;
[W89]   92:        if (ord (ch) = 8) and (length (wert)
        > 0)
[B11]   93:        then
[BH1]   94:          begin
[K78]   95:            delete (wert, length (wert), 1);
[UN3]   96:            lauf := lauf - 1;
[BU3]   97:            write (ch, ' ', ch);
[TUO]   98:          end;
[OT7]   99:        if not (ord(ch) in [8, 13, 32..127])
[B11]  100:        then
[VP2]  101:          write (bel);
[7K4]  102:      until (ord(ch) = 13);
[CB2]  103:      s := wert;
[TUO]  104:    end;
[000]  105:
[SOC]  106: {------------------\wartemeldung-----------
       --------------}
[000]  107:
[WHD]  108:    procedure wartemeldung (option : boolean)
       ;
[000]  109:
[BH1]  110:    begin
[095]  111:      if option = true
[B11]  112:      then
[66B]  113:        writexy (47, 24, 'Einen Augenblick...
       ')
[A11]  114:      else
[4D4]  115:        clreol_xy (47, 24);
[TUO]  116:    end;
[000]  117:
[7HB]  118: {------------------\hauptmenue-------------
       --------------}
[000]  119:
[247]  120:    procedure hauptmenue;
[000]  121:
[EMO]  122:      var
[DE1]  123:        i : 1..25;
[000]  124:
[BH1]  125:    begin
[T54]  126:      for i := 7 to 25 do
[7N4]  127:        clreol_xy (47,i);
[W3B]  128:      writexy (47,7, '+ und - wählen Zeichen'
       );
[5XE]  129:      writexy (47,9, 'E - Zeichen entwerfen /
        bearbeiten');
[PMB]  130:      writexy (47,10,'L - Zeichensatz laden')
       ;
[6UD]  131:      writexy (47,11,'S - Zeichensatz abspeic
       hern');
[859]  132:      writexy (47,12,'P - Probedruck');
[DRE]  133:      writexy (47,13,'D - Aktuelles Zeichen z
       um Drucker');
[E1E]  134:      writexy (47,14,'A - Alle Zeichen 33-127
        an Drucker');
[V53]  135:      normvideo;
[ANC]  136:      writexy (47,16,'Ctrl-Q  -  Programm bee
       nden');
[AZ2]  137:      lowvideo;
[TUO]  138:    end;
[000]  139:
[AKC]  140: {------------------\aktuell_anzeigen-------
       --------------}
[000]  141:
[049]  142:    procedure aktuell_anzeigen;
[000]  143:
[BH1]  144:    begin
[Z48]  145:      writexy (58,3,chr(aktuell));
[P05]  146:      writexy (64,3,'   ');
[8M3]  147:      gotoxy (64,3);
[OK4]  148:      write (aktuell);
[TUO]  149:    end;
[000]  150:
[4BC]  151: {------------------\init_eingabefeld-------
       --------------}
[000]  152:
[WW8]  153:    procedure init_eingabefeld;
```

```
[000]  154:
[EMO]  155:      var
[804]  156:        i, j : integer;
[000]  157:
[BH1]  158:    begin
[KA3]  159:      gotoxy (1,1);
[D54]  160:      for i := 1 to 24 do
[BH1]  161:        begin
[NF6]  162:          for j := 1 to breite do
[M95]  163:            write (Punkt_leer);
[9Z2]  164:          writeln;
[TUO]  165:        end;
[TUO]  166:    end;
[000]  167:
[VYB]  168: {------------------\init_fenster-----------
       --------------}
[000]  169:
[LA8]  170:    procedure init_fenster;
[000]  171:
[EMO]  172:      var
[B62]  173:        i, j : 1..37;
[000]  174:
[BH1]  175:    begin
[M54]  176:      for i := 1 to 37 do
[K54]  177:        for j := 1 to 24 do
[6Y4]  178:          fenster[i,j] := 0;
[TUO]  179:    end;
[000]  180:
[PZB]  181: {------------------\init_programm----------
       --------------}
[000]  182:
[2L8]  183:    procedure init_programm;
[000]  184:
[EMO]  185:      var
[4B4]  186:        lauf : integer;
[000]  187:
[5PE]  188: {------------------init_programm\init_zeich
       en------------}
[000]  189:
[H68]  190:      procedure init_zeichen;
[000]  191:
[EMO]  192:        var
[G53]  193:          i, j, k : 1..37;
[000]  194:
[BH1]  195:      begin
[KW4]  196:        for i := 32 to 127 do
[T54]  197:          for j := 1 to 37 do
[BY3]  198:            for k := 1 to 3 do
[OA5]  199:              zeichen[i,j,k] := 0;
[TUO]  200:      end;
[000]  201:
[PZB]  202: {------------------\init_programm----------
       --------------}
[000]  203:
[BH1]  204: begin
[NM2]  205:      clrscr;
[AZ2]  206:      lowvideo;
[B64]  207:      init_fenster;
[B84]  208:      init_zeichen;
[X04]  209:      aktuell := 33;
[024]  210:      dateiname := '';
[HH5]  211:      vorschub := true;
[667]  212:      write (lst, esc, 'p', chr(0));
[RS2]  213:      case ch of
[H42]  214:      '1' : begin
[M53]  215:              breite := 37;
[967]  216:              write (lst, esc, 'p', chr(1));
[TUO]  217:            end;
[L42]  218:      '2' : begin
[T53]  219:              breite := 29;
[232]  220:              rand := 7;
[177]  221:              write (lst, esc, 'x', chr(1));
[ZG5]  222:              write (lst, esc, 'P');
[TUO]  223:            end;
[N42]  224:      '3' : begin
[953]  225:              breite := 25;
[032]  226:              rand := 5;
[177]  227:              write (lst, esc, 'x', chr(1));
[PG5]  228:              write (lst, esc, 'M');
[TUO]  229:            end;
[R42]  230:      '4' : begin
[FX2]  231:              breite := 9;
[Y22]  232:              rand := 3;
[Y67]  233:              write (lst, esc, 'x', chr(0));
[ZG5]  234:              write (lst, esc, 'P');
[TUO]  235:            end;
[TUO]  236:      end;
[K5A]  237:      write (lst, esc, ':', chr(0), chr(0), c
       hr(0));
[8Z6]  238:      write (lst, esc, '%', chr(1));
[WF5]  239:      for lauf := 1 to 3 do
```

```
[BH1]  240:        begin
[FU4]  241:          writeln ('128': 41);
[UZ4]  242:          writeln ('64 B': 43);
[L95]  243:          writeln ('32 y': 43);
[Z85]  244:          writeln ('16 t': 43);
[U45]  245:          writeln (' 8 e': 43);
[2E4]  246:          writeln (' 4': 41);
[H86]  247:          writeln (' 2': 41, ' '.lauf);
[8Y2]  248:          if lauf = 3
[B11]  249:          then
[D93]  250:            write ('1': 41)
[A11]  251:          else
[VB4]  252:            writeln ('1': 41);
[TUO]  253:      end;
[PP5]  254:    init_eingabefeld;
[V53]  255:    normvideo;
[1NE]  256:    writexy (47,1,'FontEditor 2.0 von Gunna
       r Wilhelm');
[AZ2]  257:    lowvideo;
[4E7]  258:    writexy (47,3,'Zeichen   >');
[2ZB]  259:    write (chr(aktuell), '< Nr ', aktuell,
       ASCII');
[EB7]  260:    writexy (47,5, 'Befehl > <');
[EE3]  261:    hauptmenue;
[TUO]  262:  end;
[000]  263:
[27B]  264: {------------------\ansteuern--------------
       --------------}
[OFK]  265: { Hier wird die tatsächliche Breite des Ent
       wurfs ermit-  }
[OTL]  266: { telt, dementsprechend der Rand gesetzt un
       d schließlich }
[PXG]  267: { das Zeichen mit den erforderlichen Angabe
       n gesendet    }
[000]  268:
[F9C]  269:  procedure ansteuern (nummer : integer);
[000]  270:
[EMO]  271:    var
[12B]  272:      freiraum, links_frei, rechts_frei,
[5M7]  273:      links, rechts, a, b                :
       0..38;
[000]  274:
[BH1]  275:  begin
[WF2]  276:    links := 1;
[W85]  277:    rechts := breite;
[OD1]  278:    a := 0;
[X52]  279:    repeat
[1D1]  280:      b := 0;
[WT1]  281:      a := a + 1;
[X52]  282:      repeat
[3U1]  283:        b := b + 1;
[OT7]  284:        if zeichen[nummer,a,b] <> 0
[B11]  285:        then
[BH1]  286:          begin
[WK2]  287:            links := a;
[TX2]  288:            a := breite;
[LD1]  289:            b := 3;
[TUO]  290:          end;
[XY2]  291:      until b = 3;
[N85]  292:    until a = breite;
[000]  293:
[VH3]  294:    a := breite + 1;
[X52]  295:    repeat
[1D1]  296:      b := 0;
[WT1]  297:      a := a - 1;
[X52]  298:      repeat
[3U1]  299:        b := b + 1;
[OT7]  300:        if zeichen[nummer,a,b] <> 0
[B11]  301:        then
[BH1]  302:          begin
[733]  303:            rechts := a;
[CK2]  304:            a := links;
[LD1]  305:            b := 3;
[TUO]  306:          end;
[XY2]  307:      until b = 3;
[685]  308:    until a = links;
[8S8]  309:    zbreite := rechts - links + 1;
[MUE]  310:    write (lst, esc, '&', chr(0),chr(nummer
       ),chr(nummer));
[L89]  311:    if breite = 37     { also bei LQ PS }
[B11]  312:    then
[FLE]  313:      write (lst, chr(abstand_links), chr(r
       echts - links
[G87]  314:       + 1), chr(abstand_rechts))
[A11]  315:    else
[BH1]  316:      begin
[EV9]  317:        freiraum := breite - zbreite + rand
       ;
[X08]  318:        links_frei := freiraum div 2;
[YHB]  319:        rechts_frei := freiraum - links_fre
       i;
[44F]  320:        write (lst, chr(links_frei), chr(re
       chts - links
[1B6]  321:         + 1), chr(rechts_frei));
[TUO]  322:      end;
[BU7]  323:    for a:= links to rechts do
[YW3]  324:      for b := 1 to 3 do
[C4B]  325:        write (lst, chr(zeichen[nummer,a,b]
       ));
[TUO]  326:  end;
[000]  327:
[NHC]  328: {------------------\fenster_auslesen-------
       --------------}
[UGK]  329: { Das Punktmuster des Eingabefeldes wird au
       sgelesen und  }
[LCF]  330: { in entsprechende Byte-Werte umgewandelt.
                     }
[000]  331:
[369]  332:  procedure fenster_auslesen;
[000]  333:
[EMO]  334:    var
[KU3]  335:      spalte : 1..37;
[7W2]  336:      zeile : 0..3;
[803]  337:      punkt : 1..8;
[000]  338:
[BH1]  339:  begin
[VV5]  340:    for spalte := 1 to 37 do
[WF5]  341:      for zeile := 1 to 3 do
[X9C]  342:        zeichen [aktuell, spalte, zeile] :=
       0;
[000]  343:
[CN7]  344:    for spalte := 1 to breite do
[WF5]  345:      for zeile := 1 to 3 do
[BH6]  346:        for punkt := 8 downto 1 do
[N90]  347:          if
[3TD]  348: fenster [spalte, ((zeile - 1) * 8) + 9 - pu
       nkt] = 1
[B11]  349:           then
[BLB]  350:             zeichen [aktuell, spalte, zeile]
       :=
[PCB]  351:             zeichen [aktuell, spalte, zeile]
       +
[R67]  352:             round (exp((punkt -1) * ln (2)))
       ;
[R96]  353:    ansteuern (aktuell);
[TUO]  354:  end;
[000]  355:
[RBA]  356: {------------------\editor-----------------
       --------------}
[000]  357:
[YS5]  358:  procedure editor;
[000]  359:
[BXC]  360: {------------------editor\editormenue------
       --------------}
[000]  361:
[LM7]  362:    procedure editormenue;
[000]  363:
[EMO]  364:      var
[8E1]  365:        i : 0..25;
[000]  366:
[BH1]  367:    begin
[T54]  368:      for i := 7 to 25 do
[7N4]  369:        clreol_xy (47,i);
[PDC]  370:      writexy (47,7,'PFEILTASTEN steuern Cu
       rsor');
[RA8]  371:      writexy (47,8,'         NumLock!');
[4TE]  372:      writexy (47,10,'LEERTASTE setzt / lös
       cht Punkt');
[M1F]  373:      writexy (47,12,'Nur RETURN übernimmt
       den Entwurf');
[M6E]  374:      writexy (47,13,'   in den Anwenderzei
       chensatz');
[LVD]  375:      writexy (47,15,'^Y Entwurfsfenster lö
       schen');
[V53]  376:      normvideo;
[8BD]  377:      writexy (47,17,'^W Rückkehr zum Haupt
       menü');
[AZ2]  378:      lowvideo;
[TUO]  379:    end;
[000]  380:
[GYC]  381: {------------------editor\invertieren------
       --------------}
[000]  382:
[SN7]  383:    procedure invertieren;
[000]  384:
[BH1]  385:    begin
[YW5]  386:      if fenster[cx,cy] = 0
[B11]  387:      then
[BH1]  388:        begin
```

```
[GP5]  389:            fenster[cx,cy] := 1;
[V53]  390:            normvideo;
[1X9]  391:            writexy (cx,cy,Punkt_gesetzt);
[AZ2]  392:            lowvideo;
[YN4]  393:            gotoxy (cx,cy);
[DLO]  394:          end
[A11]  395:        else
[BH1]  396:          begin
[AC6]  397:            fenster [cx,cy] := 0;
[FY8]  398:            writexy (cx,cy, Punkt_leer);
[YN4]  399:            gotoxy (cx,cy);
[TUO]  400:          end;
[TUO]  401:      end;
[000]  402:
[9PD]  403: {------------------editor\zeige_fenster----
           --------------}
[000]  404:
[SM8]  405:      procedure zeige_fenster;
[000]  406:
[EMO]  407:        var
[P82]  408:          x, y : 1..37;
[000]  409:
[3OH]  410: {------------------editor\zeige_fenster\zei
           chen_einlesen-}
[TSH]  411: { Wandelt die Bytes (aus denen sich die Zei
           chen aufbauen)}
[1YF]  412: { um in ein Punktmuster für das Eingabefeld
                    }
[000]  413:
[HZ8]  414:        procedure zeichen_einlesen;
[000]  415:
[EMO]  416:          var
[KU3]  417:            spalte : 1..37;
[7W2]  418:            zeile : 0..3;
[F93]  419:            punkt : 1..24;
[394]  420:            puffer : byte;
[000]  421:
[BH1]  422:        begin
[VV5]  423:          for spalte := 1 to 37 do
[WF5]  424:            for zeile := 1 to 3 do
[BH1]  425:              begin
[CDD]  426:                puffer := zeichen[aktuell, sp
           alte, zeile];
[BL5]  427:                for punkt := 1 to 8 do
[BH1]  428:                begin
[9TB]  429:                fenster[spalte, (zeile * 8) +
           1 - punkt]
[KG4]  430:                 := puffer mod 2;
[LY6]  431:                puffer := puffer div 2;
[TUO]  432:                end;
[TUO]  433:              end;
[TUO]  434:        end;
[000]  435:
[9PD]  436: {------------------editor\zeige_fenster----
           --------------}
[000]  437:
[BH1]  438:      begin
[DU5]  439:        zeichen_einlesen;
[VH6]  440:        for x := 1 to breite do
[X74]  441:          for y := 1 to 24 do
[BH1]  442:            begin
[6L5]  443:              if fenster[x,y] = 1
[B11]  444:              then
[BH1]  445:                begin
[V53]  446:                  normvideo;
[G19]  447:                  writexy (x,y,Punkt_gesetzt)
           ;
[AZ2]  448:                  lowvideo;
[DLO]  449:                end
[A11]  450:              else
[XD8]  451:                writexy (x,y,Punkt_leer);
[TUO]  452:            end;
[TUO]  453:      end;
[000]  454:
[RBA]  455: {------------------\editor-----------------
           --------------}
[000]  456:
[BH1]  457:   begin
[V27]  458:      mem[0:$417] := mem [0:$417] or $20;
[9X8]  459:      {s. DOS INTERNATIONAL 8/87 Seite 73}
[XV3]  460:      editormenue;
[SR4]  461:      zeige_fenster;
[3F1]  462:      cx := 1;
[5F1]  463:      cy := 1;
[YN4]  464:      gotoxy (cx,cy);
[X52]  465:      repeat
[OH3]  466:        read (kbd,ch);
[TH4]  467:        case ord (ch) of
[EA2]  468:          52: begin
[WO2]  469:                if cx > 1
[9C4]  470:                then cx := cx - 1
[KA5]  471:                else cx := breite;
[YN4]  472:                gotoxy (cx,cy);
[TUO]  473:              end;
[AA2]  474:          50: begin
[W32]  475:                if cy < 24
[7C4]  476:                then cy := cy + 1
[AB3]  477:                else cy := 1;
[YN4]  478:                gotoxy (cx,cy);
[TUO]  479:              end;
[PA2]  480:          56: begin
[112]  481:                if cy > 1
[NC4]  482:                then cy := cy - 1
[DN3]  483:                else cy := 24;
[YN4]  484:                gotoxy (cx,cy);
[TUO]  485:              end;
[KA2]  486:          54: begin
[S44]  487:                if cx < breite
[VB4]  488:                then cx := cx + 1
[3B3]  489:                else cx := 1;
[YN4]  490:                gotoxy (cx,cy);
[TUO]  491:              end;
[3B5]  492:          32: invertieren;
[CA2]  493:          13: begin
[3O6]  494:                fenster_auslesen;
[DK9]  495:                writexy (1,25, 'Zeichenbreite
           ');
[CZ3]  496:                gotoxy (14,25);
[WV7]  497:                write (zbreite : 3, ' Spalten
           ');
[YN4]  498:                gotoxy (cx,cy);
[VP2]  499:                write (bel);
[TUO]  500:              end;
[HA2]  501:          25: begin
[PP5]  502:                init_eingabefeld;
[B64]  503:                init_fenster;
[3F1]  504:                cx := 1;
[5F1]  505:                cy := 1;
[YN4]  506:                gotoxy (cx,cy);
[TUO]  507:              end;
[TUO]  508:        end;
[MH4]  509:      until ord(ch) = 23;
[W84]  510:      clreol_xy (1,25);
[EE3]  511:      hauptmenue;
[287]  512:      mem[0:$417] := mem[0:$417] and $DF;
[9X8]  513:      {s. DOS INTERNATIONAL 8/87 Seite 73}
[TUO]  514:    end;
[000]  515:
[TEB]  516: {------------------\sende_alle-------------
           --------------}
[RVG]  517: { Mit dieser Prozedur werden die Zeichen 33
            bis 127 an    }
[OBH]  518: { den Drucker übergeben, nicht jedoch das L
           eerzeichen,    }
[W8H]  519: { welches separat (Option »D«) gesendet wer
           den muß!       }
[000]  520:
[WX6]  521:    procedure sende_alle;
[000]  522:
[EMO]  523:      var
[Z53]  524:        lauf : 33..127;
[000]  525:
[BH1]  526:    begin
[ZT5]  527:      wartemeldung (true);
[4H5]  528:      for lauf := 33 to 127 do
[YK5]  529:        ansteuern (lauf);
[VP2]  530:      write (bel);
[346]  531:      wartemeldung (false);
[TUO]  532:    end;
[000]  533:
[R6C]  534: {------------------\probeausdruck----------
           --------------}
[000]  535:
[ZL8]  536:    procedure probeausdruck;
[000]  537:
[EMO]  538:      var
[4B4]  539:        lauf : integer;
[000]  540:
[BH1]  541:    begin
[1F3]  542:      if vorschub
[B11]  543:      then
[BH1]  544:        begin
[CW5]  545:          for lauf := 1 to 25 do
[1K4]  546:            writeln (lst);
[6F5]  547:          vorschub := false;
[TUO]  548:        end;
[5W4]  549:      write (lst, fs, 'R');
[AW5]  550:      for lauf := 1 to 15 do
[1K4]  551:        writeln (lst);
[G94]  552:      write (lst,fs,'F');
```

```
[YG5]  553:      for lauf := 32 to 127 do
[SC6]  554:        write (lst, chr(lauf));
[1K4]  555:      writeln (lst);
[AW5]  556:      for lauf := 1 to 15 do
[1K4]  557:        writeln (lst);
[TUO]  558:    end;
[000]  559:
[X8A]  560: {------------------\datei------------------
           -------------}
[000]  561:
[TMB]  562:    procedure datei (option : boolean);
[2TD]  563:                      {option = true : Zeichens
           atz speichern}
[FNB]  564:                      {option = false: Zeichens
           atz laden}
[000]  565:
[EMO]  566:      var
[US4]  567:        i, j, k : integer;
[UD3]  568:        ok : boolean;
[000]  569:
[BH1]  570:    begin
[UC2]  571:      ok := true;
[X52]  572:      repeat
[ECA]  573:        writexy (47,21,'RETURN kehrt zurück')
           ;
[6W2]  574:        if option
[B11]  575:        then
[5W6]  576:          writexy (47,19,'SPEICHERN:')
[A11]  577:        else
[Y56]  578:          writexy (47,19,'LADEN:');
[BB9]  579:        writexy (47,20,'Name der Datei ?');
[O17]  580:        eingabe (11, dateiname);
[ZN4]  581:        if dateiname <> ''
[B11]  582:        then
[BH1]  583:          begin
[RY9]  584:            assign (zeichensatz, dateiname);
[6W2]  585:            if option
[B11]  586:            then
[68A]  587:              {$I-}   rewrite (zeichensatz)
           {$I+}
[A11]  588:            else
[MM8]  589:              {$I-}   reset (zeichensatz);
           {$I+}
[T55]  590:            ok := (ioresult = 0);
[8B1]  591:            if ok
[B11]  592:            then
[BH1]  593:              begin
[ZT5]  594:                wartemeldung (true);
[KW4]  595:                for i := 32 to 127 do
[T54]  596:                  for j := 1 to 37 do
[BY3]  597:                    for k := 1 to 3 do
[BH1]  598:                      begin
[6W2]  599:                        if option
[B11]  600:                        then
[RBA]  601:                          write (zeichensatz,
           zeichen[i,j,k])
[A11]  602:                        else
[F3A]  603:                          read (zeichensatz,
           zeichen[i,j,k]);
[TUO]  604:                      end;
[6N5]  605:                close (zeichensatz);
[346]  606:                wartemeldung (false);
[DLO]  607:              end
[A11]  608:            else
[BH1]  609:              begin
[6W2]  610:                if option
[B11]  611:                then
[1GC]  612:                  writexy (47,22,'IO Error (D
           iskette voll?')
[A11]  613:                else
[Y33]  614:                  writexy
[UWB]  615:                  (47,22,'IO Error (Datei nic
           ht existent?)');
[TUO]  616:              end;
[TUO]  617:        end;
[3C4]  618:      clreol_xy (47,19);
[AC4]  619:      clreol_xy (47,20);
[AC4]  620:      clreol_xy (47,21);
[ZW7]  621:    until (ok) or (dateiname = '');
[AC4]  622:    clreol_xy (47,22);
[TUO]  623:  end;
[000]  624:
[UXB]  625: {------------------\begruessung------------
           -------------}
[000]  626:
[7P7]  627:  procedure begruessung;
[000]  628:
[BH1]  629:  begin
[NM2]  630:    clrscr;
[V53]  631:    normvideo;
[81N]  632:    writeln ('Willkommen zu  F o n t E d i
           t o r  2 . 0  von Gunnar Wilhelm');
[AZ2]  633:    lowvideo;
[9Z2]  634:    writeln;
[7AR]  635:    writeln ('Dieses Programm ermöglicht es
           Ihnen, für den Drucker NEC P5/P6/P7');
[PMP]  636:    writeln ('(oder kompatible Geräte, z.B.
           Epson LQ 1500) eigene Zeichensätze');
[Y18]  637:    writeln ('zu entwerfen.');
[9Z2]  638:    writeln;
[OVE]  639:    writeln ('Stellen Sie bitte folgendes s
           icher:');
[T5T]  640:    writeln ('- Ihr Drucker ist ein 24-Nade
           l-Drucker und kompatibel zum NEC P5/P6/P7');
[U4R]  641:    writeln ('- Der Drucker ist angeschloss
           en, eingeschaltet und online!');
[VNT]  642:    writeln ('- Auf der Diskette zum Abspei
           chern von Zeichensätzen befindet sich');
[1SP]  643:    writeln ('  genügend Platz dafür (ein Z
           eichensatz = 10656 Bytes) und diese');
[G5G]  644:    writeln ('  Diskette ist nicht schreibg
           eschützt');
[9Z2]  645:    writeln;
[HER]  646:    writeln ('Für welche Schriftart möchten
           Sie eigene Zeichen entwerfen ?');
[9Z2]  647:    writeln;
[2NE]  648:    writeln ('1. Letter Quality PS      Aufl
           ösung 37 * 24');
[OAE]  649:    writeln ('2. Letter Quality 10      Aufl
           ösung 29 * 24');
[L9E]  650:    writeln ('3. Letter Quality 12      Aufl
           ösung 25 * 24');
[M5C]  651:    writeln ('4. Draft                  Aufl
           ösung  9 * 24');
[9Z2]  652:    writeln;
[RT6]  653:    write ('Ihre Wahl [1..4] = ');
[X52]  654:    repeat
[OH3]  655:      read (kbd,ch);
[2L6]  656:    until ch in ['1','2','3','4'];
[TUO]  657:  end;
[000]  658:
[ASC]  659: {==================HAUPTPROGRAMM==========
           ==============}
[000]  660:
[BH1]  661: begin
[7V3]  662:   begruessung;
[EW4]  663:   init_programm;
[X52]  664:   repeat
[HM3]  665:     gotoxy (55,5);
[FX3]  666:     read (kbd, ch);
[RS2]  667:     case ch of
[1L3]  668:       '+'     : begin
[L64]  669:                   if aktuell < 127
[KN7]  670:                   then aktuell := aktuell +
           1
[ZS4]  671:                   else aktuell := 32;
[GW5]  672:                   aktuell_anzeigen;
[TUO]  673:                 end;
[5L3]  674:       '-'     : begin
[774]  675:                   if aktuell > 32
[MN7]  676:                   then aktuell := aktuell -
           1
[XZ4]  677:                   else aktuell := 127;
[GW5]  678:                   aktuell_anzeigen;
[TUO]  679:                 end;
[AA4]  680:       'E','e' : editor;
[387]  681:       'P','p' : probeausdruck;
[FD6]  682:       'L','l' : datei(false);
[XW5]  683:       'S','s' : datei(true);
[F54]  684:       'D','d' : begin
[R96]  685:                   ansteuern (aktuell);
[VP2]  686:                   write (bel);
[TUO]  687:                 end;
[8E5]  688:       'A','a' : sende_alle;
[TM3]  689:       #17     : begin
[Y33]  690:                   writexy
[6WA]  691:                   (47,20, 'Ctrl-Q  -  Progra
           mm verlassen');
[Y33]  692:                   writexy
[TPB]  693:                   (47,21, 'Jede andere Taste
           kehrt zurück');
[CS6]  694:                   writexy (47,23, 'Wahl = ')
           ;
[FX3]  695:                   read (kbd, ch);
[AC4]  696:                   clreol_xy (47,20);
[AC4]  697:                   clreol_xy (47,21);
[AC4]  698:                   clreol_xy (47,23);
[TUO]  699:                 end;
[TUO]  700:     end;
[4U3]  701:   until ch = #17;
[NM2]  702:   clrscr;
[BV6]  703:   writeln ('Viel Spass!');
[5TO]  704: end.
```

Listing. Mit dem Font-Editor definieren Sie eigene Zeichensätze.

Software-Umschalter für Druckerkanäle

Zwei Drucker an einem Computer, dazu brauchen Sie entweder teure Hardware oder viel Geduld beim Umstecken der Kabel. Eine dritte sehr komfortable und preiswerte Lösung bietet das Turbo Pascal-Programm »lptx«.

Fast jeder, der an seinem Personalcomputer einen Drucker angeschlossen hat, macht sich wenige Gedanken darüber, über welche Schnittstelle und welchen Kanal dieser angesteuert wird. Fast alle Matrixdrucker arbeiten mit einer parallelen Schnittstelle und werden über den voreingestellten Kanal »lpt1« angesprochen. Einfach das Kabel an die entsprechenden Buchsen stecken und schon steht der Kommunikation zwischen Computer und Ausgabegeräte nichts mehr im Weg.

Verschiedene Drucker eignen sich jedoch meist auch nur optimal für verschiedene Aufgaben. Ein EDV-Listendrucker bringt meist keine Korrespondenzqualität oder sehr feine Grafikauflösung zustande. Der Druck mit einem Farbdrucker kommt für das Schreiben von einfachen Briefen zu teuer. Für verschiedene Aufgaben lohnt sich also das Anschaffen unterschiedlicher Ausgabegeräte. Anschließen kann man ja das gute Stück über den zweiten Druckerkanal »lpt2«.

Damit fangen die Probleme jedoch erst richtig an. Wer sich im guten Glauben, MS-DOS kenne ja leistungsfähige Befehle zur Kanalumschaltung, die zweite Garnitur (also Schnittstelle, Kabel und Drucker) zulegt, der sieht sich bitter enttäuscht. Weder der Mode-Befehl noch eine Standardanweisung in einer Hochsprache wie Basic oder Turbo Pascal bietet hier eine Lösung. »Mode« leitet zwar die Ausgabe auf eine serielle Schnittstelle um, nicht jedoch auf eine zweite parallele.

Behelfslösungen wie das verhaßte Herumkriechen hinter dem Computer zum Umstecken der Kabel oder auch der Kauf einer teuren Umschaltbox, kommt natürlich für einen wirklichen Software-Spezialisten nicht in Frage. Das Turbo Pascal-Programm »lptx« (Listing 1) springt bei dieser Problematik mit einer wahrhaft professionellen Lösung ein. Geben Sie es mit dem Turbo Pascal-Editor ein und compilieren Sie die Daten in eine Com-Datei (im Hauptmenü von Turbo Pascal [O] und im Optionsmenü [C] drücken – zurück wieder mit [Q]).

Nachdem Sie nun das Programm »lptx.com« auf der Diskette stehen haben, können Sie unter MS-DOS die Routine ohne Parameter aufrufen (dann wird mit einer Statusmeldung auf den jeweils anderen Druckerkanal umgeschaltet) oder mit einem Fragezeichen den aktuellen Status abfragen.

Mit

```
lptx ?
```

erfahren Sie, welcher Drucker gerade angesprochen wird und ob er ausgeschaltet, beziehungsweise Offline oder Online geschaltet ist. Besonders wenn die Drucker nicht in Sichtweite, sondern beispielsweise aus Gründen des Lärmschutzes im Nebenzimmer oder unzugänglich hinter dem Schreibtisch stehen, ist solch eine Meldung sinnvoll. Ist der Drucker nicht zur Datenübernahme bereit, also ausgeschaltet oder Offline, dann hören Sie zusätzlich einen Warnton.

Der Knackpunkt des Programms »lptx« liegt in der Prozedur »Schalter«. Die Routine arbeitet wie ein Umschalter, indem sie die Speicherstellen 8 und 10 sowie 9 und 11 bei jedem Aufruf vertauscht.

Der Druckerstatus wird mit der MS-DOS-Interrupt 17hex abgefragt und der Drucker selbst mit diesem Interrupt in den Einschaltzustand zurückgesetzt (Reset). Je nach dem Wert, der im AH-Register steht, werden folgende drei Funktionen ausgeführt:

Mit »0« wird das Zeichen im AL-Register ausgegeben, mit »1« der Druckerreset aufgerufen und mit »2« der Druckerstatus in das AH-Register eingelesen. Um den Drucker in seinen Einschaltzustand zurückzusetzen, wird die Prozedur »Drucker_Reset« aufgerufen. Softwaremäßig eingestellte Druckwerte gehen dabei verloren, per Hardware – also über Schalter – aktivierte Werte bleiben erhalten.

Damit nun auch Basic-Programmierer auf Ihre Kosten kommen, finden Sie in Listing 2 zwei Unterprogramme, die ebenfalls die Druckerports schalten. Sie sind in ein Programm eingebaut, das das Wort »Druckertest« auf dem jeweils aktiven Kanal ausgibt. Wesentlich für die Steuerung sind nur die Zeile 30 und die beiden Unterprogramme ab Zeile 1000 beziehungsweise 2000.

»lptx« zeigt, daß es nicht immer unbedingt teure Hardware sein muß, um ein scheinbar schwieriges Problem zu lösen. Eine kleine Routine hilft oft genauso weiter.

(Peter Fasse/hg)

```
[VAK]    0: program lpt_toggle; { togglen - durch einma
           ligen Aufruf }
[1Z8]    1:                     { hin und herschalten
                              }
[000]    2:
[E8G]    3: (Programm zum Umleiten der Druckausgabe  LP
           T1 <-> LPT2  )
[ZYE]    4: (written 02.08.87 by P. Fasse, Weißer Weg 4
           , 4920 Lemgo )
[GRB]    5: (Sprache : Turbo-Pascal 3.0
                              )
[000]    6:
[D7G]    7: const  segm = $40; {Segment als Constante d
           efinieren }
[000]    8:
```

```
[NF8]    9: procedure Drucker_Reset;
[1NG]   10: type ze_reg = record  {  Register initialis
           ieren     }
[NNC]   11:                ax,bx,cx,dx,bp,di,si,ds,es,flag
           s: integer;
[TUO]   12:             end;
[SB5]   13: var reg : ze_reg;
[000]   14:
[DW1]   15: begin;
[943]   16:  with reg do
[DW1]   17:   begin;
[U4D]   18:    ax:=$0100;       { Funktionscode 01 ins AH
           -Register }
[2UA]   19:                     { Funktion 01 =>Drucker r
           esetten   }
```

```
[5CC]   20:     dx:=$0000;      { Nummer des Druckers(0-2
        ) nach DX }
[7FF]   21:     intr($17,reg); { Drucker Service Routine
        aufrufen }
[TUO]   22:    end;
[TUO]   23: end;
[R2B]   24: (*----------------Ende der Procedure-------
        ---------*)
[000]   25:
[000]   26:
[FNB]   27: function Drucker_Status:integer;
[1NG]   28: type ze_reg = record  { Register initialis
        ieren     }
[NNC]   29:              ax,bx,cx,dx,bp,di,si,ds,es,flag
        s: integer;
[TUO]   30:            end;
[SB5]   31: var reg : ze_reg;
[DW1]   32: begin;
[943]   33:  with reg do
[DW1]   34:   begin;
[25D]   35:     ax:=$0200;      { Funktionscode 02 ins AH
        -Register }
[WUA]   36:                     { Funktion 02 =>Druckerst
        atus holen}
[5CC]   37:     dx:=$0000;      { Nummer des Druckers(0-2
        ) nach DX }
[7FF]   38:     intr($17,reg); { Drucker Service Routine
        aufrufen }
[746]   39:     Drucker_status:=ax;
[TUO]   40:    end;
[TUO]   41: end;
[5UA]   42: (*----------------Ende der Function--------
        ---------*)
[000]   43:
[R06]   44: procedure Schalter;
[X98]   45: var merka, merkb : integer;
[000]   46:
[DW1]   47: begin;
[PLL]   48: merka := mem [segm:8];        {fülle Variabl
        e merka mit Wert aus $40:8}
[VHL]   49:  merkb := mem [segm:9];       {fülle Variabl
        e merkb mit Wert aus $40:9}
[MSL]   50:   mem[segm:8]:=mem[segm:10]; {wechsle im Se
        gment $40 Adresse 8 mit 10}
[XSL]   51:   mem[segm:9]:=mem[segm:11]; {wechsle im Se
        gment $40 Adresse 9 mit 11}
[YWL]   52:  mem[segm:10]:=merka;         {füllen von  1
        0 mit dem alten Wert aus 8}
[ANL]   53: mem[segm:11]:=merkb;          {füllen von  1
        1 mit dem alten Wert aus 9}
[T8K]   54: end;                          {die beiden le
        tzten Zeilen der Procedure}
[1NA]   55:                               {bewirken den
        Umschalteffekt           }
[X2E]   56: (*-------------------------Ende der Procedu
        re----------------------*)
[000]   57:
[08R]   58: procedure Melde_Text;{Gibt Schalt u. Status
        zutände auf Bildschirm aus}
[716]   59: var prn: string[4];
[000]   60:
[13L]   61: begin                {Feststellen, welcher
        Drucker geschaltet ist    }
[XL6]   62:  if mem[segm:8] = 188 then
[XN2]   63:     prn := 'lpt1'
[A11]   64:   else
[313]   65:  prn := 'lpt2';
[000]   66:
[NM2]   67:  clrscr;
[AZ2]   68:    lowvideo;
[NA3]   69:  gotoxy (1,2);
[000]   70:
[2SS]   71: writeln('┌--------------------------------
        ----------------------------------------
        ┐');
[Y6V]   72: writeln('| Umschalter für parallele Drucker
        schnittstellen LPT1 <-> LPT2 (c) by P.Fasse
        |');
[OAC]   73: writeln('|
        |');
[EPS]   74: writeln('└--------------------------------
        ----------------------------------------
        ┘');
[V53]   75: normvideo;
[KLB]   76: if drucker_Status <> -28672  then begin
[Y43]   77:   gotoxy(3,4);
[KGS]   78:     writeln(#7'Aktueller Druckerport : ',pr
        n,' \ Aktueller Druckerstatus : Offline');
[DLO]   79:    end
[YS2]   80:   else begin
[Y43]   81:    gotoxy(3,4);
[O4S]   82:     writeln('Aktueller Druckerport : ',prn,
        ' \ Aktueller Druckerstatus : Online ');
[TUO]   83:    end;
[R86]   84:   gotoxy (1,5); writeln;
[AZ2]   85:  lowvideo;
[TUO]   86: end;
[X2E]   87: (*-------------------------Ende der Procedu
        re----------------------*)
[000]   88:
[PZ7]   89: begin   {Hauptprogramm}
[3W8]   90: if paramstr(1) = '?' then begin
[VB3]   91:    melde_text;
[DLO]   92:  end
[YS2]   93: else begin
[7Y2]   94:    schalter;
[VB3]   95:     melde_text;
[NW4]   96:      drucker_reset;
[DLO]   97:   end
[5TO]   98: end.
```

Listing 1. Mit »lptx« steuert Ihr PC problemlos zwei Drucker

```
[MDE] 10 REM Merker für Defaulteinstellung (LPT1) müsse
      n
[SSC] 20 REM im Hauptprogramm gesetzt werden !
[NR4] 30 MERKA = 188:MERKB = 3
[ZSA] 35 PRINT"Welcher Druckerport [ 1/2 ] ? :"
[EC3] 40 LPT$=INPUT$(1)
[HW5] 50 IF LPT$ = "1" THEN GOSUB 1000
[YW5] 60 IF LPT$ = "2" THEN GOSUB 2000
[SV6] 80 LPRINT "Druckertest"
[B91] 90 END
[GY9] 1000 REM UPROG1 - Umschaltung auf LPT1
[Y63] 1010 DEF SEG = &H40
[OM6] 1020 POKE 8,MERKA:POKE 9,MERKB
[P52] 1030 DEF SEG
[Z12] 1040 RETURN
[YY9] 2000 REM UPROG2 - Umschaltung auf LPT2
[Z63] 2010 DEF SEG = &H40
[U87] 2020 POKE 8,PEEK(10):POKE 9,PEEK(11)
[R52] 2030 DEF SEG
[022] 2040 RETURN
```

Listing 2. Zwei Unterprogramme steuern die Druckerkanäle in Basic

Inhalte der bisher erschienenen Databoxen aus DOS International

Databox 8/87
– Turbo Lister – Breakout – Grafikkurs (unter Turbo Pascal) – PC glasklar – ANSI Unterstützung für Turbo Pascal – Arithmetik mit Strings – StarWriter PC (Demo)

Databox 9/87
– Bunte Daten – SForth – Terminplaner – Grafikkurs (unter Turbo Pascal) – PC-Write (deutsche Version, Demo)

Databox 10/87
– Grafik unter Turbo Pascal – dBase II Werkstatt – RTC-Utility in GW-Basic – MS-DOS.INC (Inlude-Datei zum Aufruf von DOS Funktionen) – SForth (Teil 2) – Grafiken – (unter Turbo Pascal) – Bewegte 3D-Vektorgrafik in Turbo Pascal – Tips & Tricks für Programmierer – Demo-Version des Compilers BCi-Pascal

Databox 11/87
– Grafik unter Turbo Pascal – PC glasklar (Teil 4) – Cheddy, der Zeichensatzeditor – eine universelle Textverarbeitung in Turbo Pascal – Adressenverwaltung in Basic – Stapel-Dateien (Menüführung, Prompt, MS-DOS-Hilfe, Versteckspiel mit Dateien, Adressverwaltung) – Tips & Tricks für Programmierer – Demo-Programm zu CGX

Databox 12/87
– Listings zu den Kursen: GW-Basic, Assembler, Comal, Pascal – Batchdateien – Tips & Tricks – Demoversion der Sprache Comal

Databox 1/88
– Utilities (Kopieren mit Komfort, Dateien direkt verändert, Menüabfrage für Stapeldateien) – Tips & Tricks für Programmierer (16-Farben-Grafik für PC 1512, Sondertasten erkennen, Integerzahlen kontrolliert eingeben, Zählerroutinen als Basic-Funktion, ROM-Zeichensatz auslesen, Druckerstatus feststellen, »turbo.msg« automatisch laden) – Sämtliche Beispiele aus unseren Kursen – SForth 3.0 (erweiterte Version) – Demo-Version Quick C von Microsoft

Databox 2/88
– Listings zu den Kursen: GW-Basic, dBase, TSR-Prozesse – Basic Werkstatt (Drucker-Installation, Cursor Gestaltung) – MS-DOS-Hilfe auf Tastendruck – Hercules-Grafik mit GW-Basic – Backup-Dateien direkt lesen – Turbo Lister II –Batchdateien – Tips & Tricks

Databox 3/88
– Listings zu den Kursen: Assemblerkurs, Speicherresidente Programme selbstgemacht – Realisation eines Druckerspoolers – Hilfe aus dem Computer – Listings aus den Rubriken: Tips & Tricks, Leserforum

Die Databox zur DOS International ist erhältlich als 3 1/2"- und 5 1/4"-Diskette. Geben Sie bitte bei jeder Bestellung den Computertyp an.

DMV Daten & Medien Verlagsgesellschaft mbH
Postfach 250 · Fuldaer Straße 6 · 3440 Eschwege

Briefumschläge selbst gedruckt

Das Programm »Briefumschlag« liefert Ihnen einen kompletten Vordruck für einen Briefumschlag mit Anschrift und Absender. Wie das aussieht, bleibt allein Ihren Ideen überlassen. An Ihnen ist es, den fertigen Vordruck auszuschneiden, zu falten und zusammenzukleben.

Haben Sie schon einmal versucht, einen Briefumschlag mit Ihrem Matrixdrucker zu beschriften? Dann können Sie es sicher nachfühlen, wie groß die Enttäuschung ist, wenn auch der zehnte Versuch mit einer schrägen und verstümmelten Anschrift scheitert. Doch müßte das Ganze mit einem guten Grafikprogramm nicht hinzukriegen sein? Da es nun aber schier unmöglich ist, gekaufte Briefumschläge mit dem Drucker zu beschreiben, finden Sie hier ein Programm, das es Ihnen erlaubt, Umschläge völlig frei zu gestalten.

Das Programm Briefumschlag wurde unter GW-Basic auf einem PC mit CGA-Grafikkarte geschrieben. Das Programm läuft aber auch mit einer Herculeskarte. In diesem Fall muß aber die Grafik der höheren Auflösung angepaßt werden. Das Druckresultat ist in beiden Fällen identisch.

Das Programm beginnt zunächst mit einer Beschreibung des Arbeitsvorganges zur Herstellung eines Briefumschlags. Die rechte Bildschirmhälfte zeigt ein Blatt Endlospapier, auf dem Hilfslinien so zu sehen sind, wie sie später vom Drucker ausgegeben werden. Im linken Teil des Bildschirmes erläutert eine kurze Erklärung das Programm. Nach Drücken einer Taste wird die Fläche, die später ausgeschnitten werden muß, schraffiert (Bild 1). Mit dem nächsten Tastendruck wird der Eingabeteil gestartet. Hier verlangt das Programm die Adresse des Absenders und des Empfängers. Wollen Sie Absender oder Empfänger weglassen und später mit einem Grafikprogramm drucken, so übergehen Sie die Eingaben einfach mit der Return-Taste.

Nach der Eingabe können Sie alle Daten prüfen und gegebenenfalls korrigieren. Erst im Anschluß daran wird der Briefumschlag gedruckt. Dabei legen Sie das Papier so ein, daß sowohl die linke als auch die rechte Perforation des Endlospapiers mitbedruckt wird. Auf diese Weise erhalten Sie Umschläge in Standardgröße für Briefe im DIN A4-Format.

Das Programm »Briefumschlag« druckt dabei den Absender in Schmalschrift und Doppeldruck, während die Adresse des Empfängers in NLQ-Schrift ausgegeben wird. Ist der Druck beendet, so schneiden Sie den Bogen entsprechend Bild 1 aus, falten diesen an den markierten Stellen und kleben ihn zu einem Briefumschlag zusammen (Bild 2).

(Günter Schmidt/sp)

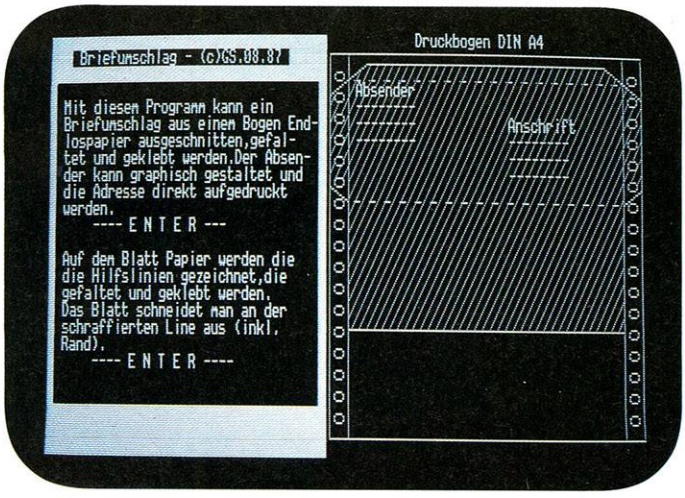


Bild 1. So sieht später der bedruckte Bogen aus

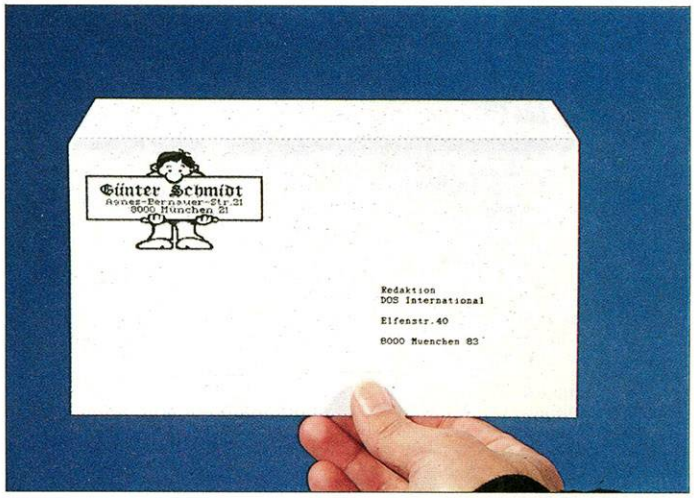


Bild 2. Dieses Kuvert entstand mit dem Programm »Briefumschlag«

```
[UV8] 10 '  -----> Briefumschlag  (c) G.S.08.87
[OB6] 20 '  =====> Hauptprogramm
[YP5] 30 SCREEN 2:KEY OFF:CLS
[843] 40 Z=2:GOSUB 430
[WD2] 50 GOSUB 130
[443] 60 Z=1:GOSUB 430
[TE5] 70 A$=INKEY$:IF A$="" THEN 70
[SE2] 80 GOSUB 680
[DD2] 90 GOTO 1130
[AN1] 100 END
[1V6] 110 ' =====> Unterprogramme
[RR9] 120 ' -----> Programmerklärung (Text)
[XA5] 130 LINE (1,198)-(296,1),1,BF
[XG9] 140 LOCATE  1,50:PRINT "Druckbogen DIN A4"
[Z6B] 150 LOCATE  2,4:PRINT" Briefumschlag - (c)GS.08.8
      7 "
[V58] 160 LOCATE  4,2:PRINT"
                "
[U5E] 170 LOCATE  5,2:PRINT" Mit diesem Programm kann e
      in      "
[XWE] 180 LOCATE  6,2:PRINT" Briefumschlag aus einem Bo
      gen End-"
```

```
[5UF] 190 LOCATE  7,2:PRINT" lospapier ausgeschnitten,
      gefal-  "
[MVD] 200 LOCATE  8,2:PRINT" tet und geklebt werden. De
      r Absen-"
[ETF] 210 LOCATE  9,2:PRINT" der kann graphisch gestalt
      et und  "
[KLE] 220 LOCATE 10,2:PRINT" die Adresse direkt aufgedr
      uckt    "
[EC9] 230 LOCATE 11,2:PRINT" werden.
                "
[MC9] 240 LOCATE 12,2:PRINT"      ---- E N T E R ---
                "
[PN5] 250 A$=INKEY$:IF A$="" THEN 250
[V78] 260 LOCATE 13,2:PRINT"
                "
[M2D] 270 LOCATE 14,2:PRINT" Auf dem Blatt Papier werde
      n       "
[8KE] 280 LOCATE 15,2:PRINT" die Hilfslinien gezeichnet
      . die   "
[C4E] 290 LOCATE 16,2:PRINT" gefaltet und geklebt werde
      n.      "
[N3E] 300 LOCATE 17,2:PRINT" Das Blatt schneidet man an
       der    "
```

```
[M8D] 310 LOCATE 18,2:PRINT" schraffierten Line aus (in
      kl.                "
[5T8] 320 LOCATE 19,2:PRINT" Rand).
                "
[VE9] 330 LOCATE 20,2:PRINT"      ---- E N T E R ----
                "
[K78] 340 LOCATE 21,2:PRINT"
                "
[T78] 350 LOCATE 22,2:PRINT"
                "
[2A7] 360 LOCATE 4,42:PRINT"Absender"
[4W7] 370 LOCATE 6,62:PRINT"Anschrift"
[XA3] 380 FOR I=1 TO 3
[8H6] 390 LOCATE 4+I,42:PRINT"--------"
[ZG6] 400 LOCATE 6+I,62:PRINT"--------"
[KT1] 410 NEXT
[BVA] 420 ' -----> Programmerklärung (Grafik)
[YD4] 430 LINE (300,10)-(630,10)
[8T4] 440 LINE (300,190)-(630,190)
[TH4] 450 LINE (300,10)-(300,190)
[DK4] 460 LINE (320,10)-(320,190)
[VK4] 470 LINE (610,10)-(610,190)
[FL4] 480 LINE (630,10)-(630,190)
[6D5] 490 FOR I=1 TO 170 STEP 10
[KD4] 500 CIRCLE (310,20+I),5
[9E4] 510 CIRCLE (620,20+I),5
[NT1] 520 NEXT
[BE4] 530 LINE (300,30)-(340,15)
[3G4] 540 LINE (630,30)-(590,15)
[KF4] 550 LINE (340,15)-(590,15)
[OR5] 560 LINE (320,23)-(610,23),1,,1000
[9S5] 570 LINE (320,80)-(610,80),1,,1000
[AF4] 580 LINE (320,80)-(300,73)
[CG4] 590 LINE (610,80)-(630,73)
[MS4] 600 LINE (320,140)-(610,140)
[N95] 610 IF Z<>1 THEN RETURN
[HZG] 620 PAINT (400,50),CHR$(1)+CHR$(2)+CHR$(4)+CHR$(8
      )+CHR$(16)+CHR$(32)+CHR$(64)+CHR$(128),2
[S25] 630 FOR I=1 TO 50 STEP 10
[6AG] 640 PAINT (302,28+I),CHR$(1)+CHR$(2)+CHR$(4)+CHR$
      (8)+CHR$(16)+CHR$(32)+CHR$(64)+CHR$(128),2
[TAG] 650 PAINT (612,28+I),CHR$(1)+CHR$(2)+CHR$(4)+CHR$
      (8)+CHR$(16)+CHR$(32)+CHR$(64)+CHR$(128),2
[YT1] 660 NEXT
[612] 670 RETURN
[AB8] 680 ' -----> Eingabe des Absenders
[CA5] 690 SCREEN 0:COLOR 14,0:CLS
[FNA] 700 LOCATE 1,30:PRINT"Eingabe des Absenders"
[B85] 710 LOCATE 3,2:COLOR 13:PRINT"Soll der Absender ü
      ber ein Graphikprogramm gedruckt werden, dann die
       Eingaben   mit ENTER übergehen."
[ZV6] 720 COLOR 10:PRINT STRING$(80,"-")
[AP9] 730 LOCATE 7,5:COLOR 15:PRINT"Vorname, Name:"
[S58] 740 LOCATE 8,5:PRINT"Straße, Nr.  :"
[698] 750 LOCATE 9,5:PRINT"PLZ, Wohnort :"
[AK7] 760 LOCATE 10,5:PRINT"Telefon     :"
[9W6] 770 COLOR 10:PRINT STRING$(80,"-")
[W48] 780 ' -----> Eingabe der Adresse
[EXC] 790 LOCATE 12,30:COLOR 14:PRINT"Eingabe der Adres
      se"
[Y99] 800 LOCATE 14,5:COLOR 15:PRINT"Anrede       :"
[E18] 810 LOCATE 15,5:PRINT"Vorname, Name:"
[UA7] 820 LOCATE 16,5:PRINT"Zusatz       :"
[EX7] 830 LOCATE 17,5:PRINT"Straße, Nr.  :"
[EB8] 840 LOCATE 18,5:PRINT"PLZ, Wohnort :"
[6W6] 850 COLOR 10:PRINT STRING$(80,"-")
[7C4] 860 ' -----> Absender
[1V9] 870 LOCATE 7,20:COLOR 12:INPUT ABNAME$:GOSUB 1120
[H29] 880 LOCATE 8,20:INPUT ABSTRASSE$:GOSUB 1120
[P59] 890 LOCATE 9,20:INPUT ABWOHNORT$:GOSUB 1120
[PX9] 900 LOCATE 10,20:LINE INPUT TELEFON$:GOSUB 1120
[9W3] 910 ' -----> Adresse
[W48] 920 LOCATE 14,20:INPUT ANREDE$:GOSUB 1120
[958] 930 LOCATE 15,20:INPUT ADNAME$:GOSUB 1120
[GA8] 940 LOCATE 16,20:INPUT ZUSATZ$:GOSUB 1120
[RL9] 950 LOCATE 17,20:INPUT ADSTRASSE$:GOSUB 1120
[HL9] 960 LOCATE 18,20:INPUT ADWOHNORT$:GOSUB 1120
[V69] 970 ' -----> Korrekturmoeglichkeit
[NS1] 980 CLS
[MU7] 990 LOCATE 5,10:COLOR 14:PRINT ABNAME$
[U17] 1000 LOCATE 7,10:PRINT ABSTRASSE$
[U17] 1010 LOCATE 9,10:PRINT ABWOHNORT$
[S47] 1020 LOCATE 11,10:PRINT TELEFON$
[CU6] 1030 LOCATE 13,50:PRINT ANREDE$
[SU6] 1040 LOCATE 15,50:PRINT ADNAME$
[K27] 1050 LOCATE 17,50:PRINT ZUSATZ$
[8D7] 1060 LOCATE 19,50:PRINT ADSTRASSE$
[7C7] 1070 LOCATE 21,50:PRINT ADWOHNORT$
[XU6] 1080 COLOR 10:PRINT STRING$(80,"-")
[D7P] 1090 LOCATE 23,2:COLOR 11:PRINT"Alle Eingaben ric
      htig? > ENTER <        -        Korrektur > K <
      :"
[6S5] 1100 A$=INKEY$:IF A$="" THEN 1100
[UU8] 1110 IF A$="k" OR A$="K" THEN 680 ELSE RETURN
[6N3] 1120 BEEP:RETURN
[C9B] 1130 '-----> Druck (Druckercode fuer DMP 3000)
[HC3] 1140 COLOR 14,0:CLS
[NU7] 1150 LOCATE 12,35:PRINT"D R U C K . . . "
[M4A] 1160 LPRINT "    ";: FOR I=1 TO 74:LPRINT "_";:NEX
      T
[ZA6] 1170 LPRINT TAB(3);"/";TAB(78);"\"
[6B6] 1180 LPRINT TAB(2);"/";TAB(79);"\"
[Z96] 1190 LPRINT TAB(1);"/";TAB(80);"\"
[P37] 1200 FOR I=1 TO 80:LPRINT ".";:NEXT
[YDD] 1210 '-----> Schmalschrift ein / Doppeldruck ein
[FG8] 1220 LPRINT CHR$(15):LPRINT CHR$(27)+"G"
[SVO] 1230 '
[G97] 1240 LPRINT:LPRINT TAB(3);ABNAME$
[FA6] 1250 LPRINT TAB(3);ABSTRASSE$
[5C6] 1260 LPRINT TAB(3);ABWOHNORT$
[CP5] 1270 LPRINT TAB(3);TELEFON$
[1ZM] 1280 '-----> Schmalschrift aus / Doppeldruck aus
      / NLQ-Standartschrift ein
[48E] 1290 LPRINT CHR$(18):LPRINT CHR$(27)+"H":LPRINT C
      HR$(27)+"x"+CHR$(1)
[HVO] 1300 '
[KW7] 1310 LPRINT:LPRINT:LPRINT:LPRINT
[BL5] 1320 LPRINT TAB(50);ANREDE$
[ZG5] 1330 LPRINT TAB(50);ADNAME$
[TU5] 1340 LPRINT TAB(50);ZUSATZ$
[FK6] 1350 LPRINT TAB(50);ADSTRASSE$
[K32] 1360 LPRINT
[WN6] 1370 LPRINT TAB(50);ADWOHNORT$
[NE9] 1375 '-----> NLQ-Standardschrift aus
[UN6] 1380 LPRINT CHR$(27)+"x"+CHR$(0)
[XWO] 1385 '
[5K6] 1390 FOR I=1 TO 5:LPRINT:NEXT
[U37] 1400 FOR I=1 TO 80:LPRINT ".";:NEXT
[MX6] 1410 FOR I=1 TO 26:LPRINT:NEXT
[047] 1420 FOR I=1 TO 80:LPRINT ".";:NEXT
```

Listing. Der Quellcode zum Programm »Briefumschlag« in GW-Basic

Wer — wo — wann?

Bei Zuschriften bitten wir Folgendes zu beachten: Haben Sie Fragen, Anregungen oder Programme, die Sie der DOS-Redaktion zukommen lassen möchten, adressieren Sie wie folgt:

Redaktion · DOS International · Elfenstr. 40 · 8000 München 83 · Tel. (089) 60 20 66

Haben Sie Fragen, die zum Beispiel Abonnements, Kleinanzeigen oder Databox-Bestellungen betreffen, wenden Sie sich bitte an den Verlag:

DMV Verlag GmbH · DOS International · Fuldaer Straße 6 · 3440 Eschwege

Tel. (0 56 51) 87 02 · Telex 993 210 dmv d · Telefax 05 65 12 07 18

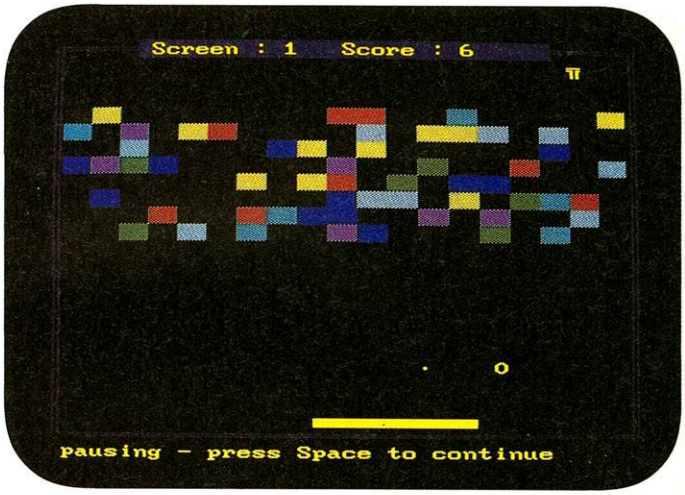


Bild 1. Spielstufe 1 ist noch leicht zu schaffen

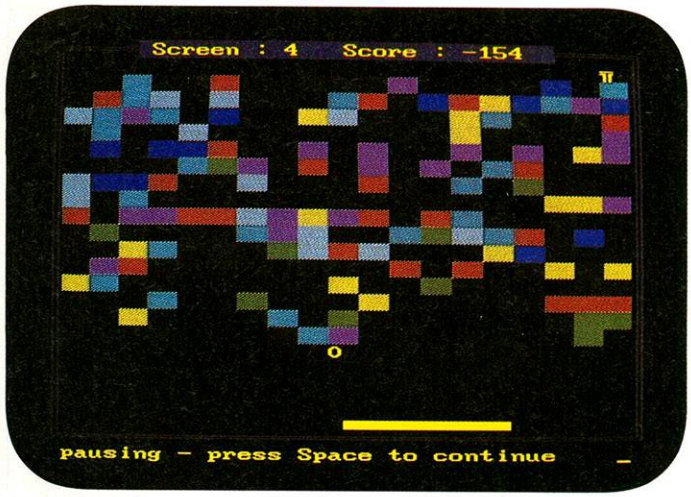


Bild 2. Hier wird's schon schwieriger

Hurricane – schnell wie der Wind

»Break Out« gehört zu den Klassikern unter den Computerspielen. Diese Turbo Pascal-Version des bekannten Spiels namens »Hurricane« verdient seinen Namen zurecht. Bei maximaler Geschwindigkeit hat der Spieler ziemlich zu kämpfen.

»Hurricane« ist dem altbekannten »Break Out« nachempfunden. Natürlich bietet das Programm einiges mehr als manch andere Spiele dieser Art. Als Programmiersprache bediente man sich Turbo Pascal, weil bei solch hoher Spielgeschwindigkeit Basic überfordert ist – zumindest ohne Compiler. »Hurricane« läuft auf PCs mit 4,77 MHz oder 8 MHz. Die Geschwindigkeitsanpassung wird in Zeile 1186 (in der Prozedur »Variablen_bewerten«) vorgenommen:

```
Verzoegerung1 := ((50 + (15 * (10 - Schwierigkeit))) DIV 4) *
Computer;
```

Das Programm wurde auf einem Schneider PC 1512 mit Farbmonitor entwickelt. Da die Bildschirmdarstellung auf Blockgrafik beruht und keine speziellen Grafikbefehle verwendet wurden, läuft »Hurricane« ausnahmslos auf allen Personalcomputern. Die Grafikausstattung spielt dabei keine Rolle.

Dem Spieler stehen 14 verschiedene Spielfelder (Bild 1 bis 4) und 10 Schwierigkeitsgrade zur Auswahl. Gesteuert wird »Hurricane« durch die Tasten [Shift], [Space], [Alt] und [Ctrl]. Die beiden Shift-Tasten bewegen den Schläger nach links oder rechts. Der Spieler ist zudem in der Lage, mit seinem Schläger Geschosse auf die Spielsteine abzufeuern. Diese Schüsse lösen Sie durch die Leertaste aus. Halten Sie die Taste ständig gedrückt, so passiert nichts, ein Dauerfeuer ist also nicht vorgesehen, Sie müssen jeden Schuß einzeln auslösen. Wollen Sie das Spiel kurz unterbrechen, weil gerade das Telefon klingelt und Sie Ihren Highscore nicht verlieren wollen, dann genügt ein Druck auf die Ctrl-Taste, und die Leertaste führt das Spiel an der unterbrochenen Stelle fort. Mit der Alt-Taste können Sie eine Spielstufe überspringen – Sie gelangen aber auf diese Weise nicht ungestraft in den nächsten Level. »Hurricane« quittiert das Überspringen einer Spielstufe mit einem Abzug von 100 Punkten.

Trifft der Ball einen massiven roten Stein, so fällt dieser langsam herunter. Wenn Sie es schaffen, ihn mit dem Schläger aufzufangen, so verändert sich die Spielsituation. Folgende Variationen können auftreten:

- Der Ball wird schneller
- Der Ball wird langsamer
- Ein zweiter Ball kommt ins Spiel
- Sie springen in die nächste Spielstufe
- Sie erhalten einen neuen Spieler

Im oberen Teil des Spielfeldes befindet sich ein »Schütze«, der kleine Steine fallen läßt. Fangen Sie diese Steinchen auf, so gibt es Zusatzpunkte. Das Programm verfügt über eine Kurzanleitung und eine Highscore-Tabelle.

(Andreas Bauer/sp)

```
[935]    0: PROGRAM Hurricane;
[000]    1:
[2U5]    2:                    (*        Hurricane
           *)
[HX6]    3:                    (*      written 1987 by
           *)
[0T6]    4:                    (*       Andreas Bauer
           *)
[LH5]    5:                    (*       8935 Fischach
           *)
[000]    6:
[EE0]    7: VAR
[Y62]    8:   i,x,k,j,
[6F0]    9:   rx,
[N94]   10:   Ball_x,Ball_y,
[WA6]   11:   Ball_xpos,Ball_ypos,
[8X2]   12:   Ball_pos,
[F75]   13:   Ball2_x,Ball2_y,
[U27]   14:   Ball2_xpos,Ball2_ypos,
[TV5]   15:   Balls_x,Balls_y,
```

```
[ZC7]   16:   Balls_xpos,Balls_ypos,
[EW7]   17:   Rakete1_xpos,Rakete1_ypos,
[TW7]   18:   Rakete2_xpos,Rakete2_ypos,
[PB8]   19:   Rakete_xpos,Rakete_ypos,
[9F2]   20:   Inhalt,
[GT5]   21:   Raketen_zaehler,
[113]   22:   Schuesse,
[0G2]   23:   Puffer1,
[7G2]   24:   Puffer2,
[0C3]   25:   Ball_Anzahl,
[XN5]   26:   Schlaegerbreite,
[TV2]   27:   Treffer,
[8U5]   28:   Level_max_Treffer,
[PC3]   29:   Stein_ypos,
[LH4]   30:   Stein_Zaehler,
[BL5]   31:   Stein_xpos,Level,
[5K7]   32:   Frequenz,Veraenderung,
[P64]   33:   Verzoegerung,
[RF4]   34:   Verzoegerung1,
[WF4]   35:   Verzoegerung2,
```

```
[KS4]   36:   Schwierigkeit,
[YR5]   37:   Schlaeger_abfrage,
[SO3]   38:   Computer,
[HX2]   39:   Speicher1,
[KX2]   40:   Speicher2,
[OU1]   41:   Taste,
[EY4]   42:   Schuetze_xpos,
[ZF3]   43:   Schuetze_x,
[6S3]   44:   Schuss_xpos,
[6S3]   45:   Schuss_ypos,
[RW5]   46:   Score                 :  INTEGER;
[X57]   47:   Schlaeger             :  STRING [20];
[2V3]   48:   a                     :  CHAR;
[2T5]   49:   aktuelle_Rakete,
[GW7]   50:   Schuetze_Zaehler      :  BYTE;
[5Y8]   51:   Feld                  :  ARRAY [1..25,1..2
        0] OF BYTE;
[KDA]   52:   Hiscore               :  ARRAY [1..10] OF
        INTEGER;
[13A]   53:   Name                  :  ARRAY [1..10] OF
        STRING [12];
[W27]   54:   Name_ein              :  STRING [12];
[WE3]   55:   Spiel_Wille,
[CA3]   56:   Zwei_Baelle,
[ZD2]   57:   Rakete,
[BY4]   58:   Rakete1,Rakete2,
[534]   59:   Stein_faellt,
[TB5]   60:   Schuss_Anzeige,
[AK2]   61:   Schuss,
[XP1]   62:   Flach,
[VA5]   63:   Loesch_Merkers,
[NY4]   64:   Loesch_Merker2,
[FT7]   65:   Loesch_Merker         :  BOOLEAN;
[000]   66:
[000]   67:
[YT4]   68: PROCEDURE Scroll;
[B21]   69: BEGIN
[D83]   70:   GotoXY (1,25);
[DZ3]   71:   For i := 1 to 26 DO
[B21]   72:   BEGIN
[9Y2]   73:     Writeln;
[FW3]   74:     k := Trunc (30 - i);
[A52]   75:     Delay (k);
[TMO]   76:   END;
[TMO]   77: END;
[000]   78:
[000]   79:
[9V4]   80: PROCEDURE Del_Line;
[B21]   81: BEGIN
[D83]   82:   GotoXY (1,25);
[9V3]   83:   Write (chr(32):39);
[TMO]   84: END;
[000]   85:
[000]   86:
[ET7]   87: PROCEDURE Instruktionen;
[B21]   88: BEGIN
[CL2]   89:   Scroll;
[XD5]   90:   TextColor (blue);
[NY2]   91:   GotoXY (1,2);
[B97]   92:   Writeln (' Hurricane');
[9Y2]   93:   Writeln;
[Y18]   94:   Writeln (' written 1987 by');
[Y38]   95:   Writeln (' Andreas Bauer');
[9Y2]   96:   Writeln;
[9Y2]   97:   Writeln;
[9Y2]   98:   Writeln;
[GT5]   99:   TextColor (yellow);
[6KD]  100:   Writeln (' Steering:  left      -> left Sh
        ift key') ;
[WCC]  101:   Writeln ('             right     -> right S
        hift key');
[5H9]  102:   Writeln ('             pausing   -> Ctrl');
[ULA]  103:   Writeln ('          next Screen -> Alt');
[6XC]  104:   Writeln ('             shooting -> Space -
        or any');
[BP9]  105:   Writeln ('                         other k
        ey');
[D83]  106:   GotoXY (1,25);
[NW4]  107:   TextColor (red);
[02B]  108:   Write ('Press any key to continue ...');
[BM7]  109:   Repeat Until Keypressed;
[CL2]  110:   Scroll;
[TMO]  111: END;
[000]  112:
[000]  113:
[5K5]  114: PROCEDURE Titelbild;
[B21]  115: BEGIN
[X42]  116:   Repeat
[B21]  117:   BEGIN
[CL2]  118:     Scroll;
[SY2]  119:     GotoXY (1,3);
[LF5]  120:     TextColor (green);
```

```
[1C8]  121:     Writeln (' * * * * * **  **  ***  **  ***
        *  * ***');
[368]  122:     Writeln (' * * * * * * * * *  *  *  * * *
        ** * *  ');
[3B8]  123:     Writeln (' *** * * **  **   *  *    ***
        **** ** ');
[D68]  124:     Writeln (' * * * * * * * * *  *  *  * * *
        * ** *  ');
[FA8]  125:     Writeln (' * * *  *  * * * * ***  **  * *
        *  * ***');
[V73]  126:     GotoXY (6,10);
[XD5]  127:     TextColor (blue);
[K1C]  128:     Write ('written 1987 by Andreas Bauer')
       ;
[XC3]  129:     FOR i := 1 TO 7 DO
[B21]  130:     BEGIN
[294]  131:       TextColor (i + 2);
[MR3]  132:       GotoXY (i,i + 13);
[F1C]  133:       Write ('Hurricane - a kind of Brick B
       uster');
[TMO]  134:     END;
[NW4]  135:     TextColor (red);
[D83]  136:     GotoXY (1,25);
[G2D]  137:     Write (' H=Hiscores - P=Play - I=Instru
       ctions');
[FN1]  138:     a := 'a';
[P9F]  139:     While (a <> 'h') AND (a <> 'p') AND (a
        <> 'i') AND (a <> ' ') DO Read (KBD,a);
[YZ1]  140:     Case a OF
[8U1]  141:     'h': BEGIN
[CL2]  142:            Scroll;
[6H3]  143:            FOR i := 1 TO 10 DO
[B21]  144:            BEGIN
[G14]  145:              TextColor (i);
[VK3]  146:              GotoXY (2,i * 2);
[6Z2]  147:              Write (i,'.');
[ZK3]  148:              GotoXY (6,i * 2);
[7A4]  149:              Write (Name[i]);
[1P3]  150:              GotoXY (30,i * 2);
[D25]  151:              Write (Hiscore [i]);
[TMO]  152:            END;
[E83]  153:            GotoXY (2,25);
[NW4]  154:            TextColor (red);
[49A]  155:            Write ('Press any Key to continu
       e');
[PR2]  156:            Read (KBD,a);
[TMO]  157:          END;
[AU1]  158:     'i': BEGIN
[OZ4]  159:            Instruktionen;
[TMO]  160:          END;
[TMO]  161:       END;
[TMO]  162:    END;
[Y85]  163:    UNTIL (a = 'p') OR (a = ' ');
[TMO]  164: END;
[000]  165:
[000]  166:
[SK7]  167: PROCEDURE Score_ausgeben;
[B21]  168: BEGIN
[1N6]  169:   TextBackGround (blue);
[GT5]  170:   TextColor (yellow);
[M73]  171:   GotoXY (20,1);
[5V3]  172:   Write (chr(32):15);
[M73]  173:   GotoXY (20,1);
[YX6]  174:   Write (' Score : ',Score);
[S67]  175:   TextBackGround (black);
[TMO]  176: END;
[000]  177:
[000]  178:
[P99]  179: PROCEDURE Ballanzahl_ausgeben;
[B21]  180: BEGIN
[NW4]  181:   TextColor (red);
[XR2]  182:   Del_Line;
[D83]  183:   GotoXY (1,25);
[3A4]  184:   Write ('Balls :');
[N83]  185:   GotoXY (9,25);
[AV6]  186:   For i := 1 TO Ball_Anzahl DO
[B21]  187:   BEGIN
[NS2]  188:     Write (' O');
[TMO]  189:   END;
[OY8]  190:   IF Schuss_Anzeige = False THEN
[B21]  191:   BEGIN
[6L3]  192:     GotoXY (30,25);
[NW4]  193:     TextColor (red);
[9A9]  194:     Write ('Level:',Schwierigkeit);
[AN7]  195:     Schuss_Anzeige := True;
[DDO]  196:   END
[ARO]  197:   ELSE
[B21]  198:   BEGIN
[6L3]  199:     GotoXY (30,25);
[GT5]  200:     TextColor (yellow);
[YW7]  201:     Write ('Shoots:',Schuesse);
[9W7]  202:     Schuss_Anzeige := False;
[TMO]  203:   END;
[TMO]  204: END;
```

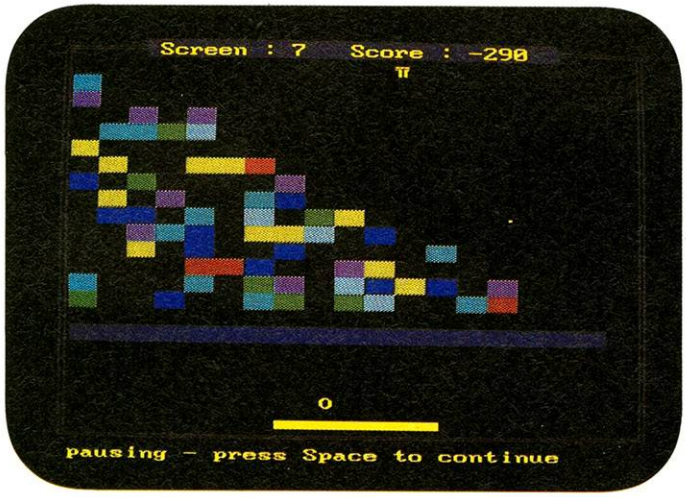


Bild 3. Spielstufe 7 ist auch nicht zu verachten

```
[000]  205:
[000]  206:
[MR7]  207: PROCEDURE Rahmen_zeichnen;
[B21]  208: BEGIN
[XD5]  209:   TextColor (blue);
[FH3]  210:   FOR i := 2 TO 23 DO
[B21]  211:   BEGIN
[U33]  212:     GotoXY (1,i);
[NK3]  213:     Write (chr(186));
[PE3]  214:     GotoXY (40,i);
[NK3]  215:     Write (chr(186));
[TMO]  216:   END;
[PH3]  217:   FOR i := 2 TO 39 DO
[B21]  218:   BEGIN
[AO3]  219:     GotoXY (i,1);
[BH3]  220:     Write (chr(205));
[1A3]  221:     GotoXY (i,24);
[BH3]  222:     Write (chr(205));
[TMO]  223:   END;
[KY2]  224:   GotoXY (1,1);
[LG3]  225:   Write (chr(201));
[P73]  226:   GotoXY (40,1);
[VK3]  227:   Write (chr(187));
[983]  228:   GotoXY (1,24);
[DG3]  229:   Write (chr(200));
[2L3]  230:   GotoXY (40,24);
[1L3]  231:   Write (chr(188));
[1N6]  232:   TextBackGround (blue);
[GT5]  233:   TextColor (yellow);
[SY2]  234:   GotoXY (7,1);
[AL8]  235:   Write (' Screen : ',Level + 1,'  ');
[S67]  236:   TextBackGround (black);
[PP6]  237:   Ballanzahl_ausgeben;
[TMO]  238: END;
[000]  239:
[000]  240:
[FD7]  241: PROCEDURE Feld_loeschen;
[B21]  242: BEGIN
[DH3]  243:   FOR i := 1 TO 25 DO
[B21]  244:   BEGIN
[RH3]  245:     FOR k := 1 TO 20 DO
[B21]  246:     BEGIN
[ZT3]  247:       Feld [k,i] := 0;
[TMO]  248:     END;
[TMO]  249:   END;
[TMO]  250: END;
[000]  251:
[000]  252:
[EP9]  253: PROCEDURE Bonus_Feld  (x,y  :    INTEGER);
[B21]  254: BEGIN
[455]  255:   IF Random (7) = 0 THEN
[B21]  256:   BEGIN
[NW4]  257:     TextColor (red);
[D14]  258:     GotoXY (x * 2,y);
[4C5]  259:     Write (chr(219),chr(219));
[4H7]  260:     Feld [x,y] := 4 + Random (10);
[TMO]  261:   END;
[TMO]  262: END;
[000]  263:
[000]  264:
[ML7]  265: PROCEDURE Level_aufbauen;
[B21]  266: BEGIN
[ML3]  267:   Treffer := 0;
[M13]  268:   CASE Level OF
```

```
[FS1]  269:  1: BEGIN
[627]  270:       Level_max_Treffer := 152;
[EH3]  271:       FOR i := 5 TO 12 DO
[B21]  272:       BEGIN
[GH3]  273:         FOR k := 1 TO 19 DO
[B21]  274:         BEGIN
[BW6]  275:           TextColor (Random (15) + 1);
[GX3]  276:           GotoXY (k * 2,i);
[GN5]  277:           Write (chr (177),chr(177));
[MU3]  278:           Feld [k,i] := 3;
[ES4]  279:           Bonus_Feld (k,i);
[TMO]  280:         END;
[TMO]  281:       END;
[TMO]  282:     END;
[GS1]  283:  2: BEGIN
[317]  284:       Level_max_Treffer := 100;
[CH3]  285:       FOR i := 4 TO 13 DO
[B21]  286:       BEGIN
[U56]  287:         FOR k := 1 + i - 4 TO 19 - i + 4 DO
[B21]  288:         BEGIN
[8W6]  289:           TextColor (Random (14) + 2);
[GX3]  290:           GotoXY (k * 2,i);
[GN5]  291:           Write (chr (177),chr(177));
[MU3]  292:           Feld [k,i] := 3;
[ES4]  293:           Bonus_Feld (k,i);
[TMO]  294:         END;
[TMO]  295:       END;
[TMO]  296:     END;
[HS1]  297:  3: BEGIN
[A17]  298:       Level_max_Treffer := 56;
[8C3]  299:       FOR i := 1 TO 4 DO
[B21]  300:       BEGIN
[GH3]  301:         FOR k := 1 TO 14 DO
[B21]  302:         BEGIN
[8W6]  303:           TextColor (Random (14) + 2);
[4D3]  304:           IF i MOD 2 = 0 THEN
[B21]  305:           BEGIN
[815]  306:             GotoXY (k * 2,i * 3 + 2);
[GN5]  307:             Write (chr (177),chr(177));
[GX4]  308:             Feld [k,i * 3 + 2] := 3;
[UU5]  309:             Bonus_Feld (k,i *3 + 2);
[DD0]  310:           END
[ARO]  311:           ELSE
[B21]  312:           BEGIN
[XL5]  313:             GotoXY (k * 2 + 10,i * 3 + 2);
[3D5]  314:             Write (chr(177),chr(177));
[YN5]  315:             Feld [k + 5,i * 3 + 2] := 3;
[476]  316:             Bonus_Feld (k + 5,i * 3 + 2);
[TMO]  317:           END;
[TMO]  318:         END;
[TMO]  319:       END;
[XD5]  320:       TextColor (blue);
[UH3]  321:       FOR k := 6 TO 19 DO
[B21]  322:       BEGIN
[CM3]  323:         GotoXY (k * 2,18);
[4C5]  324:         Write (chr(219),chr(219));
[PV3]  325:         Feld [k,18] := 2;
[TMO]  326:       END;
[TMO]  327:     END;
[KS1]  328:  4: BEGIN
[F17]  329:       Level_max_Treffer := 304;
[AH3]  330:       FOR i := 3 TO 18 DO
[B21]  331:       BEGIN
[GU3]  332:         FOR k := 1 to 19 DO
[B21]  333:         BEGIN
[8W6]  334:           TextColor (Random (14) + 2);
[GX3]  335:           GotoXY (k * 2,i);
[3D5]  336:           Write (chr(177),chr(177));
[MU3]  337:           Feld [k,i] := 3;
[ES4]  338:           Bonus_Feld (k,i);
[TMO]  339:         END;
[TMO]  340:       END;
[TMO]  341:     END;
[LS1]  342:  5: BEGIN
[D27]  343:       Level_max_Treffer := 168;
[CH3]  344:       FOR i := 4 TO 17 DO
[B21]  345:       BEGIN
[PH3]  346:         FOR k := 4 TO 15 DO
[B21]  347:         BEGIN
[8W6]  348:           TextColor (Random (14) + 2);
[GX3]  349:           GotoXY (k * 2,i);
[PC5]  350:           Write (chr(176),chr(176));
[6U3]  351:           Feld [k,i] := 1;
[TMO]  352:         END;
[TMO]  353:       END;
[TMO]  354:     END;
[MS1]  355:  6: BEGIN
[M27]  356:       Level_max_Treffer := 171;
[CH3]  357:       FOR i := 4 TO 12 DO
[B21]  358:       BEGIN
[GH3]  359:         FOR k := 1 TO 19 DO
[B21]  360:         BEGIN
[8W6]  361:           TextColor (Random (14) + 2);
```

```
[GX3]  362:            GotoXY (k * 2,i);
[3D5]  363:            Write (chr(177),chr(177));
[MU3]  364:            Feld [k,i] := 3;
[ES4]  365:            Bonus_Feld (k,i);
[TMO]  366:          END;
[TMO]  367:        END;
[XD5]  368:        TextColor (blue);
[KC3]  369:        FOR k := 1 TO 4 DO
[B21]  370:        BEGIN
[CM3]  371:          GotoXY (k * 2,18);
[4C5]  372:          Write (chr(219),chr(219));
[PV3]  373:          Feld [k,18] := 2;
[TMO]  374:        END;
[YH3]  375:        FOR k := 8 TO 19 DO
[B21]  376:        BEGIN
[CM3]  377:          GotoXY (k * 2,18);
[4C5]  378:          Write (chr(219),chr(219));
[PV3]  379:          Feld [k,18] := 2;
[TMO]  380:        END;
[TMO]  381:      END;
[NS1]  382:  7: BEGIN
[S17]  383:        Level_max_Treffer := 133;
[AH3]  384:        FOR i := 3 TO 16 DO
[B21]  385:        BEGIN
[5S3]  386:          FOR k := 1 TO i DO
[B21]  387:          BEGIN
[8W6]  388:            TextColor (Random (14) + 2);
[GX3]  389:            GotoXY (k * 2,i);
[3D5]  390:            Write (chr(177),chr(177));
[MU3]  391:            Feld [k,i] := 3;
[ES4]  392:            Bonus_Feld (k,i);
[TMO]  393:          END;
[TMO]  394:        END;
[XD5]  395:        TextColor (blue);
[6H3]  396:        FOR i := 1 TO 18 DO
[B21]  397:        BEGIN
[AM3]  398:          GotoXY (i * 2,18);
[4C5]  399:          Write (chr(219),chr(219));
[8V3]  400:          Feld [i,18] := 2;
[TMO]  401:        END;
[TMO]  402:      END;
[PS1]  403:  8: BEGIN
[S17]  404:        Level_max_Treffer := 135;
[CH3]  405:        FOR i := 4 TO 14 DO
[B21]  406:        BEGIN
[ND3]  407:          FOR k := 1 TO 9 DO
[B21]  408:          BEGIN
[8W6]  409:            TextColor (Random (14) + 2);
[UX3]  410:            GotoXY (k * 4,i);
[3D5]  411:            Write (chr(177),chr(177));
[OL4]  412:            Feld [k * 2,i] := 3;
[235]  413:            Bonus_Feld (k * 2,i);
[TMO]  414:          END;
[TMO]  415:        END;
[WT3]  416:        FOR i := 15 TO 18 DO
[B21]  417:        BEGIN
[ND3]  418:          FOR k := 1 TO 9 DO
[B21]  419:          BEGIN
[8W6]  420:            TextColor (Random (14) + 2);
[UX3]  421:            GotoXY (k * 4,i);
[PC5]  422:            Write (chr(176),chr(176));
[UK4]  423:            Feld [k * 2,i] := 1;
[TMO]  424:          END;
[TMO]  425:        END;
[TMO]  426:      END;
[RS1]  427:  9: BEGIN
[R17]  428:        Level_max_Treffer := 228;
[AH3]  429:        FOR i := 3 TO 14 DO
[B21]  430:        BEGIN
[GH3]  431:          FOR k := 1 TO 19 DO
[B21]  432:          BEGIN
[8W6]  433:            TextColor (Random (14) + 2);
[GX3]  434:            GotoXY (k * 2,i);
[PC5]  435:            Write (chr(176),chr(176));
[6U3]  436:            Feld [k,i] := 1;
[TMO]  437:          END;
[TMO]  438:        END;
[TMO]  439:      END;
[6R1]  440: 10: BEGIN
[317]  441:        Level_max_Treffer := 100;
[ES3]  442:        FOR j := 1 TO 100 DO
[B21]  443:        BEGIN
[8W6]  444:          TextColor (Random (14) + 2);
[XG1]  445:          REPEAT
[A54]  446:            i := Random (14) + 3;
[764]  447:            k := Random (19) + 1;
[PS4]  448:          UNTIL Feld [k,i] = 0;
[GX3]  449:          GotoXY (k * 2,i);
[3D5]  450:          Write (chr(177),chr(177));
[MU3]  451:          Feld [k,i] := 3;
[ES4]  452:          Bonus_Feld (k,i);
[TMO]  453:        END;
[TMO]  454:      END;
```

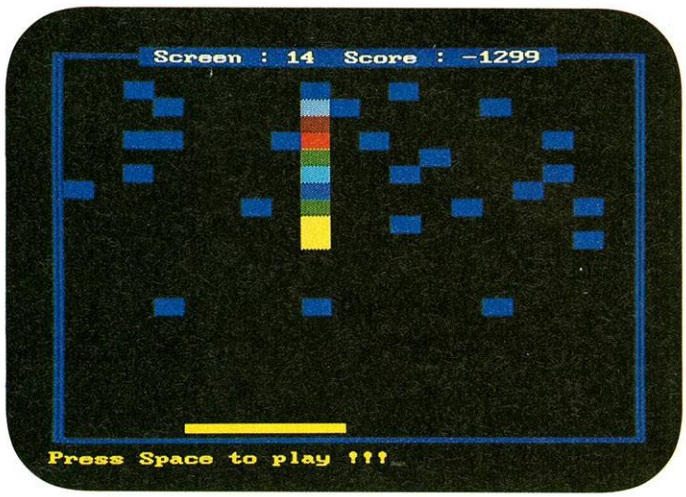


Bild 4. Die weißen Steine können nicht abgeschossen werden

```
[8R1]  455: 11: BEGIN
[G27]  456:       Level_max_Treffer := 98;
[MC3]  457:       FOR i := 0 TO 6 DO
[B21]  458:       BEGIN
[GV4]  459:         FOR k := 8 - i TO 8 + i DO
[B21]  460:         BEGIN
[8W6]  461:           TextColor (Random (14) + 2);
[TR4]  462:           GotoXY (k * 2 + 4,i + 3);
[3D5]  463:           Write (chr(177),chr(177));
[965]  464:           Feld [k + 2,i + 3] := 3;
[NK5]  465:           Bonus_Feld (k + 2,i + 3);
[TMO]  466:         END;
[TMO]  467:       END;
[PH3]  468:       FOR k := 4 TO 16 DO
[B21]  469:       BEGIN
[8W6]  470:         TextColor (Random (14) + 2);
[CM3]  471:         GotoXY (k * 2,10);
[3D5]  472:         Write (chr(177),chr(177));
[6V3]  473:         Feld [k,10] := 3;
[AF4]  474:         Bonus_Feld (k,10);
[TMO]  475:       END;
[MH3]  476:       FOR i := 8 TO 13 DO
[B21]  477:       BEGIN
[ZO5]  478:         FOR k := - 2 + i TO 24 - i DO
[B21]  479:         BEGIN
[8W6]  480:           TextColor (Random (14) + 2);
[5S4]  481:           GotoXY (k * 2 - 2,i + 3);
[3D5]  482:           Write (chr(177),chr(177));
[865]  483:           Feld [k - 1,i + 3] := 3;
[2L5]  484:           Bonus_Feld (k - 1,i + 3);
[TMO]  485:         END;
[TMO]  486:       END;
[TMO]  487:     END;
[AR1]  488: 12: BEGIN
[527]  489:       Level_max_Treffer := 240;
[AH3]  490:       FOR i := 3 TO 17 DO
[B21]  491:       BEGIN
[KH3]  492:         FOR k := 2 TO 18 DO
[B21]  493:         BEGIN
[8W6]  494:           TextColor (Random (14) + 2);
[GX3]  495:           GotoXY (k * 2,i);
[3D5]  496:           Write (chr(177),chr(177));
[MU3]  497:           Feld [k,i] := 3;
[ES4]  498:           Bonus_Feld (k,i);
[TMO]  499:         END;
[TMO]  500:       END;
[AH3]  501:       FOR i := 3 TO 17 DO
[B21]  502:       BEGIN
[ME3]  503:         GotoXY (20,i);
[AV4]  504:         Write (chr(32),chr(32));
[XH3]  505:         Feld [10,i] := 0;
[TMO]  506:       END;
[TMO]  507:     END;
[CR1]  508: 13: BEGIN
[A17]  509:       Level_max_Treffer := 112;
[AZ3]  510:       For i := 3 to 18 DO
[B21]  511:       BEGIN
[GK3]  512:         For k := 2 TO 8 DO
[B21]  513:         BEGIN
[8W6]  514:           TextColor (Random (14) + 2);
[4D3]  515:           IF i MOD 2 = 0 THEN
[B21]  516:           BEGIN
[N84]  517:             GotoXY (k * 4 + 2,i);
[3D5]  518:             Write (chr(177),chr(177));
[2Y4]  519:             Feld [k * 2 + 1,i] := 3;
[WU5]  520:             Bonus_Feld (k * 2 + 1,i);
[DDO]  521:           END
```

```
[ARO]  522:            ELSE
[B21]  523:            BEGIN
[UX3]  524:              GotoXY (k * 4,i);
[3D5]  525:              Write (chr(177),chr(177));
[OL4]  526:              Feld [k * 2,i] := 3;
[235]  527:              Bonus_Feld (k * 2,i);
[TMO]  528:            END;
[TMO]  529:          END;
[TMO]  530:        END;
[TMO]  531:      END;
[ER1]  532: 14: BEGIN
[Z27]  533:        Level_max_Treffer := 8;
[CH3]  534:        FOR i := 4 TO 12 DO
[B21]  535:        BEGIN
[8W6]  536:          TextColor (Random (14) + 2);
[3F3]  537:          GotoXY (18,i);
[3D5]  538:          Write (chr(177),chr(177));
[PF3]  539:          Feld [9,i] := 3;
[MG4]  540:          Bonus_Feld (9,i);
[TMO]  541:        END;
[KH3]  542:        FOR j := 1 TO 25 DO
[B21]  543:        BEGIN
[XD5]  544:          TextColor (blue);
[XG1]  545:          REPEAT
[A54]  546:            i := Random (14) + 3;
[764]  547:            k := Random (19) + 1;
[PS4]  548:          UNTIL Feld [k,i] = 0;
[GX3]  549:          GotoXY (k * 2,i);
[4C5]  550:          Write (chr(219),chr(219));
[DU3]  551:          Feld [k,i] := 2;
[TMO]  552:        END;
[TMO]  553:      END;
[TMO]  554:    END;
[TMO]  555: END;
[000]  556:
[000]  557:
[9BC]  558: PROCEDURE Ton_ausgeben (Frequenz : INTEGER)
       ;
[B21]  559: BEGIN
[N45]  560:   Sound (Frequenz);
[Y92]  561:   Delay (10);
[UV2]  562:   NoSound;
[TMO]  563: END;
[000]  564:
[000]  565:
[MV8]  566: PROCEDURE Schlaeger_ausgeben;
[B21]  567: BEGIN
[BK3]  568:   GotoXY (x-1,23);
[GT5]  569:   TextColor (yellow);
[RY4]  570:   Write (Schlaeger);
[XH2]  571:   IF x = 2 THEN
[B21]  572:   BEGIN
[XD5]  573:     TextColor (blue);
[583]  574:     GotoXY (1,23);
[NK3]  575:     Write (chr(186));
[TMO]  576:   END;
[VG8]  577:   IF x = 39 - Schlaegerbreite THEN
[B21]  578:   BEGIN
[XD5]  579:     TextColor (blue);
[XK3]  580:     GotoXY (40,23);
[NK3]  581:     Write (chr(186));
[TMO]  582:   END;
[TMO]  583: END;
[000]  584:
[000]  585:
[BZ7]  586: PROCEDURE Raketen_abfragen;
[B21]  587: BEGIN
[WMA]  588:   GotoXY (Rakete_xpos,Rakete_ypos);
[PD3]  589:   Write (Chr(32));
[ZT8]  590:   Rakete_ypos := Rakete_ypos - 1;
[XZC]  591:   IF aktuelle_Rakete = 1 THEN Rakete_xpos :
       = x;
[RPK]  592:   IF aktuelle_Rakete = 2 THEN Rakete_xpos :
       = x + Schlaegerbreite;
[7ME]  593:   Inhalt := Feld [Rakete_xpos DIV 2,Rakete_
       ypos];
[H1A]  594:   IF (Inhalt = 0) AND (Rakete_ypos > 1) THE
       N
[B21]  595:   BEGIN
[WMA]  596:     GotoXY (Rakete_xpos,Rakete_ypos);
[NW4]  597:     TextColor (red);
[KK3]  598:     Write (chr(239));
[DDO]  599:   END
[ARO]  600:   ELSE
[B21]  601:   BEGIN
[FY4]  602:     Rakete := False;
[EL3]  603:     CASE Inhalt OF
[ES1]  604:      0: BEGIN
[O84]  605:           Frequenz := 300;
[5D7]  606:           Ton_ausgeben (Frequenz);
[TMO]  607:         END;
[FS1]  608:      1: BEGIN
[O84]  609:           Frequenz := 300;
[5D7]  610:           Ton_ausgeben (Frequenz);
[2PB]  611:           Feld [Rakete_xpos DIV 2,Rakete_ypos
       ] := 3;
[L6F]  612:           GotoXY (Rakete_xpos - Rakete_xpos M
       OD 2,Rakete_ypos);
[9R6]  613:           TextColor (Random(14) + 2);
[3D5]  614:           Write (chr(177),chr(177));
[OYC]  615:           Bonus_Feld (Rakete_xpos DIV 2,Raket
       e_ypos);
[TMO]  616:         END;
[GS1]  617:      2: BEGIN
[O84]  618:           Frequenz := 300;
[5D7]  619:           Ton_ausgeben (Frequenz);
[TMO]  620:         END;
[TMO]  621:       END;
[634]  622:       IF Inhalt >= 3 THEN
[B21]  623:       BEGIN
[ZNB]  624:         Feld [Rakete_xpos DIV 2,Rakete_ypos]
       := 0;
[L6F]  625:         GotoXY (Rakete_xpos - Rakete_xpos MOD
       2,Rakete_ypos);
[AV4]  626:         Write (chr(32),chr(32));
[RR6]  627:         Treffer := Treffer + 1;
[O84]  628:         Frequenz := 300;
[5D7]  629:         Ton_ausgeben (Frequenz);
[TMO]  630:       END;
[TMO]  631:     END;
[TMO]  632: END;
[000]  633:
[000]  634:
[XS7]  635: PROCEDURE Raketen_bewegen;
[B21]  636: BEGIN
[6A4]  637:   IF Rakete1 THEN
[B21]  638:   BEGIN
[1H6]  639:     aktuelle_Rakete := 1;
[GR4]  640:     Rakete := True;
[WT7]  641:     Rakete_xpos := Rakete1_xpos;
[WT7]  642:     Rakete_ypos := Rakete1_ypos;
[TR5]  643:     Raketen_abfragen;
[H58]  644:     Rakete1_xpos := Rakete_xpos;
[K58]  645:     Rakete1_ypos := Rakete_ypos;
[325]  646:     Rakete1 := Rakete;
[TMO]  647:   END;
[8A4]  648:   IF Rakete2 THEN
[B21]  649:   BEGIN
[5H6]  650:     aktuelle_Rakete := 2;
[GR4]  651:     Rakete := True;
[2U7]  652:     Rakete_xpos := Rakete2_xpos;
[2U7]  653:     Rakete_ypos := Rakete2_ypos;
[TR5]  654:     Raketen_abfragen;
[S58]  655:     Rakete2_xpos := Rakete_xpos;
[T58]  656:     Rakete2_ypos := Rakete_ypos;
[A25]  657:     Rakete2 := Rakete;
[TMO]  658:   END;
[WVG]  659:   IF Rakete1 OR Rakete2 THEN Rakete := True
       ELSE Rakete := False;
[TMO]  660: END;
[000]  661:
[000]  662:
[T55]  663: PROCEDURE Schuetze;
[B21]  664: BEGIN
[LED]  665:   IF (Schuetze_xpos = 2) OR (Schuetze_xpos
       = 39) THEN
[B21]  666:   BEGIN
[VO8]  667:     Schuetze_x := - Schuetze_x;
[TMO]  668:   END;
[KK7]  669:   GotoXY (Schuetze_xpos,2);
[PD3]  670:   Write (chr(32));
[ZAE]  671:   Schuetze_xpos := Schuetze_xpos + Schuetze
       _x;
[GT5]  672:   TextColor (yellow);
[KK7]  673:   GotoXY (Schuetze_xpos,2);
[ZG3]  674:   Write (chr(203));
[D83]  675:   GotoXY (1,25);
[BLB]  676:   IF (Random (20) = 1) AND (Schuss = False)
       THEN
[B21]  677:   BEGIN
[VV4]  678:     Schuss := True;
[1M4]  679:     Schuss_ypos := 3;
[7Z8]  680:     Schuss_xpos := Schuetze_xpos;
[YTA]  681:     GotoXY (Schuss_xpos,Schuss_ypos);
[GT5]  682:     TextColor (yellow);
[S36]  683:     TextBackGround (red);
[RK3]  684:     Write (chr(249));
[S67]  685:     TextBackGround (black);
[TMO]  686:   END;
[FO4]  687:   IF Schuss THEN
[B21]  688:   BEGIN
[YTA]  689:     GotoXY (Schuss_xpos,Schuss_ypos);
[PD3]  690:     Write (chr(32));
[WLE]  691:     IF Feld [Schuss_xpos DIV 2,Schuss_ypos]
       <> 0 THEN
```

```
[B21]  692:      BEGIN
[8TC]  693:        CASE  Feld [Schuss_xpos DIV 2,Schuss_
       ypos] OF
[FS1]  694:          1: BEGIN
[8W6]  695:               TextColor (Random (14) + 2);
[2KF]  696:               GotoXY (Schuss_xpos - Schuss_xpo
       s MOD 2,Schuss_ypos);
[PC5]  697:               Write (chr(176),chr(176));
[TMO]  698:             END;
[GS1]  699:          2: BEGIN
[2KF]  700:               GotoXY (Schuss_xpos - Schuss_xpo
       s MOD 2,Schuss_ypos);
[XD5]  701:               TextColor (blue);
[4C5]  702:               Write (chr(219),chr(219));
[TMO]  703:             END;
[TMO]  704:          END;
[8ME]  705:        IF Feld [Schuss_xpos DIV 2,Schuss_ypo
       s] >= 3 THEN
[B21]  706:        BEGIN
[LME]  707:          IF Feld [Schuss_xpos DIV 2,Schuss_y
       pos] = 3 THEN
[B21]  708:          BEGIN
[2KF]  709:            GotoXY (Schuss_xpos - Schuss_xpos
        MOD 2,Schuss_ypos);
[ZS6]  710:            TextColor (Random (14)+ 2);
[3D5]  711:            Write (chr(177),chr(177));
[DDO]  712:          END
[ARO]  713:          ELSE
[B21]  714:          BEGIN
[2KF]  715:            GotoXY (Schuss_xpos - Schuss_xpos
        MOD 2,Schuss_ypos);
[NW4]  716:            TextColor (red);
[4C5]  717:            Write (chr(219),chr(219));
[TMO]  718:          END;
[TMO]  719:        END;
[TMO]  720:      END;
[G19]  721:      Schuss_ypos := Schuss_ypos + 1;
[YTA]  722:      GotoXY (Schuss_xpos,Schuss_ypos);
[GT5]  723:      TextColor (yellow);
[S36]  724:      TextBackGround (red);
[RK3]  725:      Write (chr(249));
[S67]  726:      TextBackGround (black);
[D83]  727:      GotoXY (1,25);
[OS6]  728:      IF Schuss_ypos = 23 THEN
[B21]  729:      BEGIN
[U25]  730:        Schuss := False;
[YTA]  731:        GotoXY (Schuss_xpos,Schuss_ypos);
[PD3]  732:        Write (chr(32));
[WPK]  733:        IF (Schuss_xpos <= x + Schlaegerbreit
       e) AND (Schuss_xpos >= x) THEN
[B21]  734:        BEGIN
[6X4]  735:          Score := Score + 50;
[A95]  736:          Score_ausgeben;
[YTA]  737:          GotoXY (Schuss_xpos,Schuss_ypos);
[GT5]  738:          TextColor (yellow);
[RG3]  739:          Write (chr(220));
[DDO]  740:        END
[ARO]  741:        ELSE
[B21]  742:        BEGIN
[6X4]  743:          Score := Score - 50;
[A95]  744:          Score_ausgeben;
[TMO]  745:        END;
[TMO]  746:      END;
[TMO]  747:    END;
[TMO]  748: END;
[000]  749:
[000]  750:
[WU8]  751: PROCEDURE Schlaeger_bewegung;
[B21]  752: BEGIN
[ZD2]  753:   Taste := 0;
[GX4]  754:   CASE Mem [$0000:$0417] OF
[FS1]  755:     1: BEGIN
[1E2]  756:          Taste := 1;
[TMO]  757:        END;
[GS1]  758:     2: BEGIN
[7M2]  759:          Taste := -1;
[TMO]  760:        END;
[KS1]  761:     4: BEGIN
[D83]  762:          GotoXY (1,25);
[ZRD]  763:          Write (' pausing - press Space to
       continue    ');
[FN1]  764:          a := 'a';
[ZE3]  765:          While a <> ' ' DO
[B21]  766:          BEGIN
[PR2]  767:            Read (KBD,a);
[TMO]  768:          END;
[D83]  769:          GotoXY (1,25);
[9V3]  770:          Write (chr(32):39);
[PP6]  771:          Ballanzahl_ausgeben;
[TMO]  772:        END;
```

```
[PS1]  773:     8: BEGIN
[UD9]  774:          Treffer := Level_max_Treffer;
[T45]  775:          Score := Score - 100;
[D83]  776:          GotoXY (1,25);
[9V3]  777:          Write (chr(32):39);
[GT5]  778:          TextColor (yellow);
[D83]  779:          GotoXY (1,25);
[D3A]  780:          Write ('You are in the next Level
       ');
[PR2]  781:          Delay (5000);
[TMO]  782:        END;
[TMO]  783:   END;
[T25]  784:   IF Keypressed THEN
[B21]  785:   BEGIN
[WB5]  786:     IF (Schuesse > 0) THEN
[B21]  787:     BEGIN
[906]  788:       IF Rakete = False THEN
[B21]  789:       BEGIN
[2U4]  790:         Rakete1_ypos := 22;
[8R4]  791:         Rakete1_xpos := x;
[9U4]  792:         Rakete2_ypos := 22;
[7CA]  793:         Rakete2_xpos := x + Schlaegerbreite
       ;
[AW6]  794:         Schuesse := Schuesse - 1;
[3S8]  795:         IF Stein_faellt = False THEN
[B21]  796:         BEGIN
[6L3]  797:           GotoXY (30,25);
[CP3]  798:           Write (chr(32):9);
[6L3]  799:           GotoXY (30,25);
[GT5]  800:           TextColor (yellow);
[YW7]  801:           Write ('Shoots:',Schuesse);
[TMO]  802:         END;
[105]  803:         Rakete1 := True;
[805]  804:         Rakete2 := True;
[GR4]  805:         Rakete := True;
[TMO]  806:       END;
[TMO]  807:     END;
[TMO]  808:   END;
[Z6C]  809:   IF (x < 39 - Schlaegerbreite) AND (Taste
       = 1) THEN
[B21]  810:   BEGIN
[O54]  811:     x := x + Taste;
[U46]  812:     Schlaeger_ausgeben;
[DDO]  813:   END
[WG9]  814:   ELSE IF Taste = 1 THEN Delay (Puffer1);
[347]  815:   IF (x > 2) AND (Taste =-1) THEN
[B21]  816:   BEGIN
[O54]  817:     x := x + Taste;
[U46]  818:     Schlaeger_ausgeben;
[DDO]  819:   END
[1FA]  820:   ELSE IF Taste = - 1 THEN Delay (Puffer1);
[9S9]  821:   IF Taste = 0 THEN Delay (Puffer2);
[DXU]  822:   IF Raketen_zaehler < 3 THEN Raketen_zaehl
       er := Raketen_zaehler + 1 ELSE Raketen_zaehl
       er := 0;
[FDG]  823:   IF Rakete AND (Raketen_zaehler = 0) THEN
       Raketen_bewegen;
[TMO]  824: END;
[000]  825:
[000]  826:
[RFA]  827: PROCEDURE Schlaegerbreite_aendern;
[B21]  828: BEGIN
[E04]  829:   Schlaeger := ' ';
[V59]  830:   FOR i := 1 to Schlaegerbreite + 1 DO
[B21]  831:   BEGIN
[509]  832:     Schlaeger := Schlaeger + chr(220);
[TMO]  833:   END;
[ZN7]  834:   Schlaeger := Schlaeger + ' ';
[TMO]  835: END;
[000]  836:
[000]  837:
[OH7]  838: PROCEDURE Ball_verloren;
[B21]  839: BEGIN
[L48]  840:   Ball_Anzahl := Ball_Anzahl -1;
[683]  841:   GotoXY (2,23);
[8V3]  842:   Write (chr(32):38);
[GB1]  843:   x := 10;
[U46]  844:   Schlaeger_ausgeben;
[2V8]  845:   GotoXY (Ball_xpos,Ball_ypos);
[PD3]  846:   Write (chr(32));
[ZV2]  847:   Ball_x := 1;
[S33]  848:   Ball_y := -1;
[RB4]  849:   Ball_xpos := 3;
[S94]  850:   Ball_ypos := 22;
[XR2]  851:   Del_Line;
[NL6]  852:   IF Ball_Anzahl > 0 THEN
[B21]  853:   BEGIN
[8H3]  854:     For i := 1 TO 4 DO
[B21]  855:     BEGIN
[D83]  856:       GotoXY (1,25);
[NW4]  857:       TextColor (red);
[L79]  858:       Write ('start in ',4-i,' seconds');
```

```
[BG2]  859:        Sound (400);
[Y92]  860:        Delay (10);
[UV2]  861:        NoSound;
[PR2]  862:        Delay (1000);
[TMO]  863:      END;
[TMO]  864:    END;
[PP6]  865:    Ballanzahl_ausgeben;
[TMO]  866: END;
[OOO]  867:
[OOO]  868:
[HE7]  869: PROCEDURE Stein_am_Boden;
[B21]  870: BEGIN
[XR2]  871:    Del_Line;
[PP6]  872:    Ballanzahl_ausgeben;
[TMO]  873: END;
[OOO]  874:
[OOO]  875:
[547]  876: PROCEDURE Ball_bewegen;
[B21]  877: BEGIN
[2V8]  878:    GotoXY (Ball_xpos,Ball_ypos);
[TDC]  879:    IF Loesch_Merker = False THEN Write (chr(
       32))
[Y87]  880:    ELSE Loesch_Merker := False;
[ZZ9]  881:    Ball_xpos := Ball_xpos + Ball_x;
[YC3]  882:    IF Flach THEN
[B21]  883:    BEGIN
[63G]  884:      IF (Ball_xpos = 2) OR (Ball_xpos = 39)
       THEN Ball_x := - Ball_x;
[ZZ9]  885:      Ball_xpos := Ball_xpos + Ball_x;
[TMO]  886:    END;
[FOA]  887:    Ball_ypos := Ball_ypos + Ball_y;
[2V8]  888:    GotoXY (Ball_xpos,Ball_ypos);
[GT5]  889:    TextColor (yellow);
[9F3]  890:    Write (chr(79));
[S18]  891:    IF Stein_faellt = True THEN
[B21]  892:    BEGIN
[W9A]  893:      Stein_Zaehler := Stein_Zaehler + 1;
[7Y6]  894:      IF Stein_Zaehler = 4 THEN
[B21]  895:      BEGIN
[K55]  896:        Stein_Zaehler := 0;
[P3A]  897:        GotoXY (Stein_xpos,Stein_ypos);
[AV4]  898:        Write (chr(32),chr(32));
[30E]  899:        IF Feld [Stein_xpos DIV 2,Stein_ypos]
        <> 0 THEN
[B21]  900:        BEGIN
[31C]  901:          CASE  Feld [Stein_xpos DIV 2,Stein_
       ypos] OF
[FS1]  902:            1: BEGIN
[8W6]  903:                 TextColor (Random (14) + 2);
[P3A]  904:                 GotoXY (Stein_xpos,Stein_ypos)
       ;
[PC5]  905:                 Write (chr(176),chr(176));
[TMO]  906:               END;
[GS1]  907:            2: BEGIN
[P3A]  908:                 GotoXY (Stein_xpos,Stein_ypos)
       ;
[XD5]  909:                 TextColor (blue);
[4C5]  910:                 Write (chr(219),chr(219));
[TMO]  911:               END;
[TMO]  912:           END;
[ROE]  913:         IF Feld [Stein_xpos DIV 2,Stein_ypo
       s] >= 3 THEN
[B21]  914:           BEGIN
[ELD]  915:             IF Feld [Stein_xpos DIV 2,Stein_y
       pos] = 3 THEN
[B21]  916:               BEGIN
[P3A]  917:                 GotoXY (Stein_xpos,Stein_ypos);
[ZS6]  918:                 TextColor (Random (14)+ 2);
[3D5]  919:                 Write (chr(177),chr(177));
[DDO]  920:               END
[ARO]  921:               ELSE
[B21]  922:               BEGIN
[P3A]  923:                 GotoXY (Stein_xpos,Stein_ypos);
[NW4]  924:                 TextColor (red);
[4C5]  925:                 Write (chr(219),chr(219));
[TMO]  926:               END;
[TMO]  927:           END;
[TMO]  928:       END;
[MA8]  929:       Stein_ypos := Stein_ypos + 1;
[P3A]  930:       GotoXY (Stein_xpos,Stein_ypos);
[NW4]  931:       TextColor (red);
[3D5]  932:       Write (chr(177),chr(177));
[TL6]  933:       IF Stein_ypos = 23 THEN
[B21]  934:       BEGIN
[Z75]  935:         Stein_am_Boden;
[K55]  936:         Stein_Zaehler := 0;
[3N6]  937:         Stein_faellt := False;
[MP6]  938:         GotoXY (Stein_xpos,23);
[AV4]  939:         Write (chr(32),chr(32));
[U46]  940:         Schlaeger_ausgeben;
[M1K]  941:         IF ((Stein_xpos >= x) AND (Stein_xp
       os <= x + Schlaegerbreite)) THEN
[B21]  942:         BEGIN
```

```
[084]  943:           Frequenz := 300;
[5D7]  944:           Ton_ausgeben (Frequenz);
[A84]  945:           Frequenz := 500;
[5D7]  946:           Ton_ausgeben (Frequenz);
[T45]  947:           Score := Score + 100;
[A95]  948:           Score_ausgeben;
[2P5]  949:           CASE Veraenderung OF
[FS1]  950:           1: BEGIN
[R4N]  951:                IF Verzoegerung = Verzoegeru
       ng1 THEN Verzoegerung := Verzoegerung2
[LBA]  952:                ELSE Verzoegerung := Verzoeg
       erung2;
[TMO]  953:              END;
[GS1]  954:           2: BEGIN
[9G7]  955:                IF Schlaegerbreite < 15 THEN
[B21]  956:                BEGIN
[P15]  957:                  IF x >= 3 THEN x := x - 2;
[6DC]  958:                  Schlaegerbreite := Schlaeg
       erbreite + 2;
[9L8]  959:                  Schlaegerbreite_aendern;
[U46]  960:                  Schlaeger_ausgeben;
[TMO]  961:                END;
[TMO]  962:              END;
[HS1]  963:           3: BEGIN
[737]  964:                IF Schlaegerbreite > 5 THEN
[B21]  965:                BEGIN
[EDC]  966:                  Schlaegerbreite := Schlaeg
       erbreite - 2;
[9L8]  967:                  Schlaegerbreite_aendern;
[683]  968:                  GotoXY (2,23);
[8V3]  969:                  Write (chr(32):38);
[U46]  970:                  Schlaeger_ausgeben;
[TMO]  971:                END;
[TMO]  972:              END;
[KS1]  973:           4: BEGIN
[SG6]  974:                IF Ball_anzahl < 16 THEN
[B21]  975:                BEGIN
[6X4]  976:                  Score := Score - 50;
[A95]  977:                  Score_ausgeben;
[XR2]  978:                  Del_Line;
[GT5]  979:                  TextColor (yellow);
[D83]  980:                  GotoXY (1,25);
[G09]  981:                  Write ('You got a Bonusbal
       l !!!');
[PR2]  982:                  Delay (5000);
[RC8]  983:                  Ball_Anzahl := Ball_Anzahl
        + 1;
[XR2]  984:                  Del_Line;
[PP6]  985:                  Ballanzahl_ausgeben;
[TMO]  986:                END;
[TMO]  987:              END;
[LS1]  988:           5: BEGIN
[UD9]  989:                Treffer := Level_max_Treffer
       ;
[XR2]  990:                Del_Line;
[GT5]  991:                TextColor (yellow);
[D83]  992:                GotoXY (1,25);
[2EA]  993:                Write ('You are in the next
       Level !!!');
[T45]  994:                Score := Score + 100;
[A95]  995:                Score_ausgeben;
[PR2]  996:                Delay (5000);
[XR2]  997:                Del_Line;
[PP6]  998:                Ballanzahl_ausgeben;
[TMO]  999:              END;
[MS1] 1000:           6: BEGIN
[2H7] 1001:                IF Zwei_Baelle = False THEN
[B21] 1002:                BEGIN
[FX4] 1003:                  Score := Score - 25;
[A95] 1004:                  Score_ausgeben;
[734] 1005:                  Ball2_xpos := 3;
[254] 1006:                  Ball2_ypos := 22;
[LB3] 1007:                  Ball2_x := 1;
[UM3] 1008:                  Ball2_y := -1;
[305] 1009:                  Zwei_Baelle := True;
[TMO] 1010:                END;
[TMO] 1011:              END;
[NS1] 1012:           7: BEGIN
[YW4] 1013:                Score := Score + 10;
[A95] 1014:                Score_ausgeben;
[AWE] 1015:                IF Schlaeger_abfrage = 2 THE
       N Schlaeger_abfrage := 6
[GN7] 1016:                ELSE Schlaeger_abfrage := 2;
[TMO] 1017:              END;
[PS1] 1018:           8: BEGIN
[OX4] 1019:                Score := Score + 20;
[A95] 1020:                Score_ausgeben;
[14F] 1021:                IF Flach = True THEN Flach :
       = False ELSE Flach := True;
[TMO] 1022:              END;
[RS1] 1023:           9: BEGIN
[YW4] 1024:                Score := Score - 10;
[A95] 1025:                Score_ausgeben;
[2V8] 1026:                GotoXY (Ball_xpos,Ball_ypos)
```

```
[PD3] 1027:                Write (chr(32));
[S33] 1028:                Ball_y := -1;
[RB4] 1029:                Ball_ypos := 2;
[TMO] 1030:              END;
[6R1] 1031:           10: BEGIN
[OX4] 1032:                Score := Score - 20;
[A95] 1033:                Score_ausgeben;
[D83] 1034:                GotoXY (1,25);
[9V3] 1035:                Write (chr(32):39);
[GT5] 1036:                TextColor (yellow);
[D83] 1037:                GotoXY (1,25);
[P89] 1038:                Write ('You got some bullets
      !!!');
[PR2] 1039:                Delay (5000);
[XR2] 1040:                Del_Line;
[A3A] 1041:                Schuesse := Schuesse + Rando
      m (20) + 5;
[PP6] 1042:                Ballanzahl_ausgeben;
[TMO] 1043:              END;
[TMO] 1044:            END;
[TMO] 1045:          END;
[TMO] 1046:        END;
[TMO] 1047:      END;
[TMO] 1048:    END;
[5CC] 1049:    Inhalt := Feld [Ball_xpos DIV 2,Ball_ypos
      ];
[134] 1050:    IF Inhalt > 0 THEN
[B21] 1051:    BEGIN
[EL3] 1052:      CASE Inhalt OF
[FS1] 1053:      1: BEGIN
[3X9] 1054:           Score := Score + Schwierigkeit * 3
      ;
[A95] 1055:           Score_ausgeben;
[T84] 1056:           Frequenz := 800;
[5D7] 1057:           Ton_ausgeben (Frequenz);
[CTA] 1058:           Feld [Ball_xpos DIV 2,Ball_ypos] :
      = 3;
[EYD] 1059:           GotoXY (Ball_xpos - Ball_xpos MOD
      2,Ball_ypos);
[8W6] 1060:           TextColor (Random (14) + 2);
[3D5] 1061:           Write (chr(177),chr(177));
[YGB] 1062:           Bonus_Feld (Ball_xpos DIV 2,Ball_y
      pos);
[5Z4] 1063:           Ball_y := -Ball_y;
[SC6] 1064:           Loesch_Merker := TRUE;
[TMO] 1065:         END;
[GS1] 1066:      2: BEGIN
[7X9] 1067:           Score := Score + Schwierigkeit * 5
      ;
[A95] 1068:           Score_ausgeben;
[P74] 1069:           Frequenz := 100;
[5D7] 1070:           Ton_ausgeben (Frequenz);
[EYD] 1071:           GotoXY (Ball_xpos - Ball_xpos MOD
      2,Ball_ypos);
[XD5] 1072:           TextColor (blue);
[4C5] 1073:           Write (chr(219),chr(219));
[5Z4] 1074:           Ball_y := -Ball_y;
[F0A] 1075:           Ball_ypos := Ball_ypos + Ball_y;
[TMO] 1076:         END;
[TMO] 1077:       END;
[634] 1078:       IF Inhalt >= 3 THEN
[B21] 1079:       BEGIN
[XSA] 1080:         Feld [Ball_xpos DIV 2,Ball_ypos] :=
      0;
[EYD] 1081:         GotoXY (Ball_xpos - Ball_xpos MOD 2,
      Ball_ypos);
[AV4] 1082:         Write (chr(32),chr(32));
[2W8] 1083:         GotoXY (ball_xpos,Ball_ypos);
[9F3] 1084:         Write (chr(79));
[9T9] 1085:         Score := Score + Schwierigkeit;
[RR6] 1086:         Treffer := Treffer + 1;
[A95] 1087:         Score_ausgeben;
[M84] 1088:         Frequenz := 700;
[5D7] 1089:         Ton_ausgeben (Frequenz);
[G34] 1090:         IF Inhalt > 3 THEN
[B21] 1091:         BEGIN
[3S8] 1092:           IF Stein_faellt = False THEN
[B21] 1093:           BEGIN
[DK7] 1094:             Stein_ypos := Ball_ypos;
[W9C] 1095:             Stein_xpos := Ball_xpos - Ball_x
      pos MOD 2;
[K55] 1096:             Stein_Zaehler := 0;
[246] 1097:             Stein_faellt := True;
[D83] 1098:             GotoXY (1,25);
[9V3] 1099:             Write (chr(32):39);
[977] 1100:             TextColor (red + blink);
[D83] 1101:             GotoXY (1,25);
[SF7] 1102:             Veraenderung := Inhalt - 3;
[2P5] 1103:             CASE Veraenderung OF
[LCD] 1104:           1: Write (' This stone change',chr(
      39),'s speed');
[KTE] 1105:           2: Write (' This stone change',chr(
      39),'s your board');
[LTE] 1106:           3: Write (' This stone change',chr(
      39),'s your board');
[1GF] 1107:           4: Write (' This stone award',chr(3
      9),'s you a bonus ball');
[SMH] 1108:           5: Write (' This stone show',chr(39
      ),'s you the next screen');
[G9E] 1109:           6: Write (' This stone double',chr(
      39),'s your ball');
[4MF] 1110:           7: Write (' This stone change',chr(
      39),'s boards speed');
[B7G] 1111:           8: Write (' This stone change',chr(
      39),'s the flight angle');
[VCH] 1112:           9: Write (' This put',chr(39),'s yo
      ur ball into the topline');
[DHG] 1113:          10: Write (' This stone award',chr(3
      9),'s you some bullets');
[TMO] 1114:             END;
[TMO] 1115:           END;
[TMO] 1116:         END;
[5Z4] 1117:         Ball_y := -Ball_y;
[TMO] 1118:       END;
[TMO] 1119:     END;
[D83] 1120:     GotoXY (1,25);
[ZAA] 1121:     IF (Ball_xpos = 2) OR (Ball_xpos = 39) TH
      EN
[B21] 1122:     BEGIN
[YY4] 1123:       Ball_x := -Ball_x;
[584] 1124:       Frequenz := 400;
[5D7] 1125:       Ton_ausgeben (Frequenz);
[TMO] 1126:     END;
[Z9A] 1127:     IF (Ball_ypos = 2) OR (Ball_ypos = 22) TH
      EN
[B21] 1128:     BEGIN
[O84] 1129:       Frequenz := 300;
[5D7] 1130:       Ton_ausgeben (Frequenz);
[5E9] 1131:       Ball_pos := Ball_x + Ball_xpos;
[5Z4] 1132:       Ball_y := -Ball_y;
[816] 1133:       IF (Ball_ypos = 22) THEN
[B21] 1134:       BEGIN
[1LF] 1135:         IF ((Ball_pos < x) OR (Ball_pos > x +
      Schlaegerbreite)) THEN
[B21] 1136:         BEGIN
[695] 1137:           IF Zwei_Baelle THEN
[B21] 1138:           BEGIN
[TE5] 1139:             Zwei_Baelle := False;
[GP9] 1140:             GotoXY (Ball2_xpos,Ball2_ypos);
[PD3] 1141:             Write (chr(32));
[2V8] 1142:             GotoXY (Ball_xpos,Ball_ypos);
[PD3] 1143:             Write (chr(32));
[K33] 1144:             Ball_x := -1;
[S33] 1145:             Ball_y := -1;
[RB4] 1146:             Ball_xpos := 3;
[S94] 1147:             Ball_ypos := 22;
[DDO] 1148:           END
[ARO] 1149:           ELSE
[B21] 1150:           BEGIN
[H94] 1151:             Frequenz := 1000;
[PS4] 1152:             Ball_verloren;
[5D7] 1153:             Ton_ausgeben (Frequenz);
[TMO] 1154:           END;
[TMO] 1155:         END;
[2GE] 1156:         IF ((Ball_pos = x) AND (Ball_x = 1))
      THEN Ball_x := -Ball_x;
[5LM] 1157:         IF ((Ball_pos = x + Schlaegerbreite)
      AND (Ball_x = -1)) THEN Ball_x := -Ball_x;
[TMO] 1158:       END;
[TMO] 1159:     END;
[TMO] 1160: END;
[OOO] 1161:
[OOO] 1162:
[OW8] 1163: PROCEDURE Variablen_bewerten;
[B21] 1164: BEGIN
[GB1] 1165:   x := 10;
[ZV2] 1166:   Ball_x := 1;
[S33] 1167:   Ball_y := -1;
[RB4] 1168:   Ball_xpos := 3;
[S94] 1169:   Ball_ypos := 22;
[LK5] 1170:   Schuetze_xpos := 3;
[BG4] 1171:   Schuetze_x := 1;
[K55] 1172:   Stein_Zaehler := 0;
[6V6] 1173:   Schuetze_Zaehler := 0;
[3N6] 1174:   Stein_faellt := False;
[HZ6] 1175:   Schlaeger_abfrage := 2;
[ZU4] 1176:   IF Computer = 3 THEN
[B21] 1177:   BEGIN
[MH3] 1178:     Puffer1 := 12;
[OK3] 1179:     Puffer2 := 13;
[DDO] 1180:   END
[ARO] 1181:   ELSE
[B21] 1182:   BEGIN
[B83] 1183:     Puffer1 := 4;
[K83] 1184:     Puffer2 := 4;
[TMO] 1185:   END;
[ADH] 1186:   Verzoegerung1 := ((50 + (15 * (10 - Schwi
      erigkeit))) DIV 4) * Computer;
```

```
[TLA] 1187:    Verzoegerung2 := Verzoegerung1 DIV 2;
[KF9] 1188:    Verzoegerung := Verzoegerung1;
[05D] 1189:    Schlaegerbreite := 10 - (Schwierigkeit DI
          V 2);
[4C7] 1190:    Loesch_Merker := False;
[5F7] 1191:    Loesch_Merker2 := False;
[TE5] 1192:    Zwei_Baelle := False;
[FY4] 1193:    Rakete := False;
[VY4] 1194:    Rakete1 := False;
[2Z4] 1195:    Rakete2 := False;
[Y44] 1196:    Speicher1 := 1;
[554] 1197:    Speicher2 := 2;
[4F4] 1198:    Flach := False;
[U25] 1199:    Schuss := False;
[AN7] 1200:    Schuss_Anzeige := True;
[P87] 1201:    Schuesse := Random (40) + 20;
[TMO] 1202: END;
[000] 1203:
[000] 1204:
[B21] 1205: BEGIN
[S77] 1206:    TextBackground (black);
[NF2] 1207:    ClrScr;
[T54] 1208:    TextMode (C40);
[CL2] 1209:    Scroll;
[R73] 1210:    GotoXY (2,10);
[ZDA] 1211:    Write ('Schneider PC (8086/8 Mhz) (y/n) ?
          ');
[FN1] 1212:    a := 'a';
[H1D] 1213:    While (a <> 'y') AND (a <> 'n') AND (a <>
          ' ') DO Read (KBD,a);
[YP1] 1214:    CASE a OF
[G84] 1215:    ' ': Computer := 3;
[2E4] 1216:    'y': Computer := 3;
[8D4] 1217:    'n': Computer := 1;
[TMO] 1218:    END;
[L45] 1219:    Spiel_Wille := True;
[6H3] 1220:    FOR i := 1 TO 10 DO
[B21] 1221:    BEGIN
[6B6] 1222:      Hiscore [i] := 110 - i * 10;
[AZ4] 1223:      Name [i] := 'PC 1512';
[TMO] 1224:    END;
[DP5] 1225:    While Spiel_Wille DO
[B21] 1226:    BEGIN
[R13] 1227:      Titelbild;
[CL2] 1228:      Scroll;
[NW4] 1229:      TextColor (red);
[083] 1230:      GotoXY (3,12);
[42A] 1231:      Write ('Which Skill Level (1 to 10) ?')
          ;
[983] 1232:      GotoXY (4,14);
[XD5] 1233:      TextColor (blue);
[R9A] 1234:      Write ('(10 is the most difficult)');
[XK3] 1235:      GotoXY (35,12);
[LC5] 1236:      Schwierigkeit := 1;
[8X5] 1237:      Read (Schwierigkeit);
[FLD] 1238:      IF Schwierigkeit > 10 THEN Schwierigkei
          t := 10;
[4BD] 1239:      IF Schwierigkeit < 1 THEN Schwierigkeit
          := 1;
[CL2] 1240:      Scroll;
[ED2] 1241:      Score := 0;
[FD2] 1242:      Level := 0;
[V74] 1243:      Ball_Anzahl := 5;
[4B6] 1244:      While (Ball_Anzahl > 0) DO
[B21] 1245:      BEGIN
[M44] 1246:        IF Level = 14 THEN
[B21] 1247:        BEGIN
[FD2] 1248:          Level := 0;
[UNK] 1249:          IF Schwierigkeit < 9 THEN Schwierig
          keit := Schwierigkeit + 1;
[TMO] 1250:        END;
[Y66] 1251:        Variablen_bewerten;
[CL2] 1252:        Scroll;
[CN5] 1253:        Rahmen_zeichnen;
[A95] 1254:        Score_ausgeben;
[9L8] 1255:        Schlaegerbreite_aendern;
[U46] 1256:        Schlaeger_ausgeben;
[HP4] 1257:        Level := Level + 1;
[XP4] 1258:        Feld_loeschen;
[MA5] 1259:        Level_aufbauen;
[XR2] 1260:        Del_Line;
[D83] 1261:        GotoXY (1,25);
[GT5] 1262:        TextColor (yellow);
[818] 1263:        Write ('Press Space to play !!!');
[RU1] 1264:        a := 'y';
[ZE3] 1265:        While a <> ' ' DO
[B21] 1266:        BEGIN
[4L3] 1267:          i := Random (100);
[PR2] 1268:          Read (KBD,a);
[TMO] 1269:        END;
[CP7] 1270:        Speicher1 := Mem [$0000:$041C];
```

```
[PY6] 1271:        Speicher2 := Speicher1;
[D83] 1272:        GotoXY (1,25);
[9V3] 1273:        Write (chr(32):39);
[PP6] 1274:        Ballanzahl_ausgeben;
[000] 1275:
[XYH] 1276:        While ((Treffer < Level_max_Treffer)
          AND (Ball_Anzahl > 0)) DO
[B21] 1277:        BEGIN
[FBG] 1278:          For j := 1 TO Schlaeger_abfrage DO
          Schlaeger_bewegung;
[314] 1279:          Ball_bewegen;
[695] 1280:          IF Zwei_Baelle THEN
[B21] 1281:          BEGIN
[4H5] 1282:            Balls_x := Ball_x;
[CH5] 1283:            Balls_y := Ball_y;
[5K7] 1284:            Balls_xpos := Ball_xpos;
[GK7] 1285:            Balls_ypos := Ball_ypos;
[7SA] 1286:            Loesch_Merkers := Loesch_Merker;
[UT4] 1287:            Ball_x := Ball2_x;
[1U4] 1288:            Ball_y := Ball2_y;
[RM7] 1289:            Ball_xpos := Ball2_xpos;
[1N7] 1290:            Ball_ypos := Ball2_ypos;
[97A] 1291:            Loesch_Merker := Loesch_Merker2;
[314] 1292:            Ball_bewegen;
[B87] 1293:            Ball2_ypos := Ball_ypos;
[O87] 1294:            Ball2_xpos := Ball_xpos;
[775] 1295:            Ball2_y := Ball_y;
[Z65] 1296:            Ball2_x := Ball_x;
[1CA] 1297:            Loesch_Merker2 := Loesch_Merker;
[185] 1298:            Ball_x := Balls_x;
[885] 1299:            Ball_y := Balls_y;
[TS7] 1300:            Ball_xpos := Balls_xpos;
[3T7] 1301:            Ball_ypos := Balls_ypos;
[GNA] 1302:            Loesch_Merker := Loesch_Merkers;
[TMO] 1303:          END;
[YDC] 1304:          Schuetze_Zaehler := Schuetze_Zaehle
          r + 1;
[VX7] 1305:          IF Schuetze_Zaehler >= 3 THEN
[B21] 1306:          BEGIN
[6V6] 1307:            Schuetze_Zaehler := 0;
[733] 1308:            Schuetze;
[TMO] 1309:          END;
[NZ5] 1310:          Delay (Verzoegerung);
[A6G] 1311:          FOR i := 1 TO Schlaeger_abfrage DO
          Schlaeger_bewegung;
[TMO] 1312:        END;
[TMO] 1313:      END;
[CL2] 1314:      Scroll;
[HK3] 1315:      GotoXY (15,10);
[XD5] 1316:      TextColor (blue);
[PD4] 1317:      Write ('GAME OVER');
[RL7] 1318:      IF Score >= Hiscore [10] THEN
[B21] 1319:      BEGIN
[LF5] 1320:        TextColor (green);
[483] 1321:        GotoXY (7,12);
[8G9] 1322:        Write ('You have got a Hiscore !!');
[C83] 1323:        GotoXY (7,14);
[FR9] 1324:        Write ('Please enter your name : ');
[M83] 1325:        GotoXY (7,16);
[185] 1326:        Readln (Name_ein);
[1B1] 1327:        i := 10;
[XXA] 1328:        While ((Score >= Hiscore [i]) AND (i
          > 1)) DO
[B21] 1329:        BEGIN
[MV1] 1330:          i := i - 1;
[TMO] 1331:        END;
[ZN9] 1332:        IF Score < Hiscore [1] THEN i := i +
          1;
[VX4] 1333:        FOR x := 1 TO 11 - i DO
[B21] 1334:        BEGIN
[412] 1335:          k := 10 - x;
[T59] 1336:          Hiscore [k + 1] := Hiscore [k];
[8C6] 1337:          Name [k + 1] := Name [k];
[TMO] 1338:        END;
[ZS5] 1339:        Hiscore [i] := Score;
[1V5] 1340:        Name [i] := Name_ein;
[TMO] 1341:      END;
[FL3] 1342:      GotoXY (10,18);
[NW4] 1343:      TextColor (red);
[AG8] 1344:      Write ('Another Game (y/n) ?');
[FN1] 1345:      a := 'a';
[GCA] 1346:      While ((a <> 'y') AND (a <> 'n')) DO Re
          ad (KBD,a);
[XL8] 1347:      IF a = 'n' THEN Spiel_Wille := False
[TMO] 1348:    END;
[5LO] 1349: END.
```

Listing. Der Turbo Pascal-Code für »Hurricane«

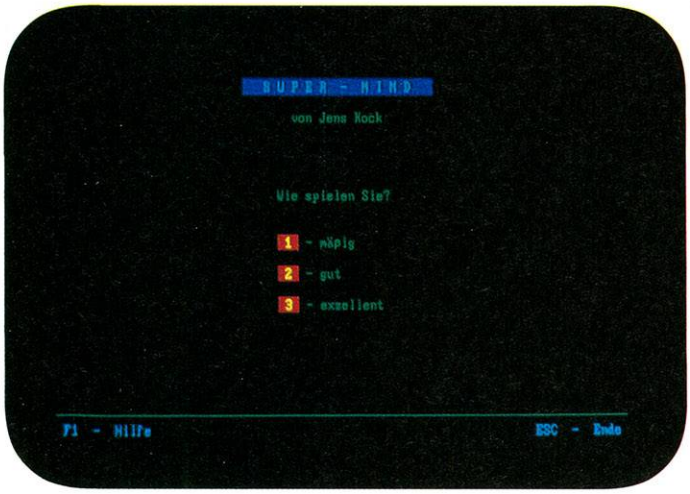


Bild 1. Das Menü von »Super-Mind« gibt Ihnen die Auswahl zwischen drei Schwierigkeitsgraden

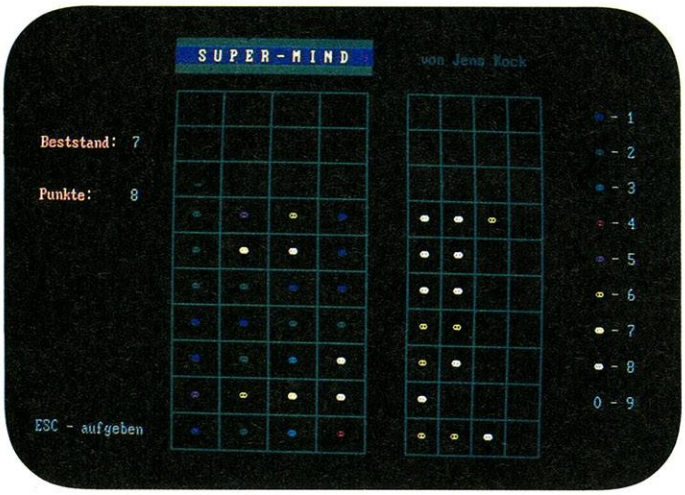


Bild 2. Das Super-Mind-Spielfeld

Super-Mind,
die Master-Mind-Version für Ihren PC

Dieses Turbo Pascal-Programm ist eine sehr gelungene Version des bekannten Kombinationsspieles Master-Mind. Das Denkspiel läuft problemlos auf allen Personalcomputern mit Farbgrafikkarte.

Sicher kennen Sie das Knobelspiel Master-Mind. Es gilt, eine vierstellige Farbkombination zu erraten, die ein Mitspieler willkürlich zusammengestellt hat. Super-Mind ist die Umsetzung auf Ihren PC, wobei der Computer die Rolle des Gegenspielers übernimmt. Ihr PC wählt vier beliebige Farben aus, die Sie erraten, beziehungsweise kombinieren müssen. Für jede Farbe, bei der auch die Position stimmt, zeigt Super-Mind einen braunen Punkt an. Stimmt nur die Farbe, aber steht sie an der falschen Position, so wird ein weißer Punkt gesetzt.

Nach dem Start wird zuerst das Menü (Bild 1) auf dem Bildschirm ausgegeben. In ihm bestimmen Sie den Schwierigkeitsgrad und können die Spielanleitung studieren. [Esc] beendet Super-Mind in jedem Spielabschnitt. Um ein Versehen auszuschließen, versichert sich Super-Mind natürlich vor dem entgültigen Abbruch, ob Sie wirklich nicht mehr weiterspielen wollen.

Die Schwierigkeitsgrade beziehen sich auf die Farbkombinationen. Im Schierigkeitsgrad 1 sind lediglich solche Farbkombinationen zugelassen, in denen eine Farbe maximal einmal vorkommt, im mittleren Schwierigkeitsgrad dürfen sich Farben auch mehrmals wiederholen, und im schwierigsten Spielmodus sind auch Leerstellen in der Farbkombination erlaubt.

Die wichtigsten Felder in dem Turbo Pascal-Programm enthalten die Farbkombinationen (»comb«), die Eingabe des Spielers (»move«) und die Auswertung der Eingabe (»Marks«). Die Abfrage der Tastatur geschieht über die sogenannten »Scancodes« (Tastaturcodes).

Der Code wird über einen MS-DOS-Aufruf im Register »AX« abgelegt. Von dort aus wird die Eingabe ausgewertet. Da den Funktionstasten jeweils zwei Codes zugeordnet sind (Code 1 ist allerdings immer Null), muß für diesen Fall der MS-DOS-Aufruf wiederholt werden. Wollen Sie mehr über diese »Scancodes« erfahren, so ziehen Sie bitte Ihr MS-DOS-Handbuch zu Rate.

Der Quellcode des Super-Mind-Spieles (Listing) enthält durchgehend Kommentare für den Benutzer. Dadurch ist das Verständnis gewährleisten. Auf diese Weise können Sie leicht selbständig Änderungen an dem Programm vornehmen.

Für den Aufbau des Spielfeldes (Bild 2) sorgt die Prozedur »DrawBoard«. Beachten Sie beim Abtippen die Lage der Grafikzeichen peinlichst genau, da sonst der Aufbau des Bildschirms nicht stimmt.

Die Prozedur »Evaluate« wertet den Zug des Spielers aus. Für jede richtige Farbe an richtiger Stelle wird das Bewertungsfeld »marks[place]« auf eins gesetzt. Stimmt zwar die Farbe, nicht aber der Ort, so erhält »marks[place]« den Wert zwei.

Das Hauptprogramm von Super-Mind besteht zum großen Teil aus ineinandergeschachtelten Schleifen. Die Hauptschleife umschließt ein ganzes Spiel, während die inneren Schleifen für einzelne Züge oder Unterroutinen verantwortlich sind. Zuletzt wird das Ergebnis ausgegeben. Der Computer teilt Ihnen die Bewertung des letzten Spiels mit – mit einer kleinen Melodie untermalt. Nach Spielende wird das Menü wieder aufgerufen. Von hier aus können Sie das Spiel erneut starten und vielleicht einen höheren Schwierigkeitsgrad wählen.

(Jens Kock/sp)

```
[UV5]    0: Program SuperMind;
[000]    1:
[000]    2:
[1YK]    3: { von: Jens Kock, Postfach 5070, 9000 Windh
           oek, S.W.Afrika/Namibia    }
[3MN]    4: { Dieses Programm ist die Computerversion d
           es Spieles "Master-Mind". }
[DNL]    5: { In der Prozedur "Help" stehen nähere Inst
           ruktionen.                 }
[000]    6:
[000]    7:
[6K8]    8: Var count,n,difficulty,code,
[G88]    9:     score,hiscore              :Integer;
[D46]   10:     entry                      :Char;
[ZFD]   11:     comb,move,marks            :Array [1..4]
           Of Integer;
[SB5]   12:     x,y                        :Byte;
[6M8]   13:     solution,game              :Boolean;
[000]   14:
[000]   15:
[XEP]   16: Procedure ScanKeyboard;                { der
           letzte Tastendruck wird in 'entry' }
[ER6]   17:                                        { abge
           legt.                                   }
[GX6]   18:   Type register = Record
[7Z9]   19:                    AX,BX,CX,DX,BP,SI,DI,DS
           ,ES,FLAGS:Integer;
[TTO]   20:                  End;
[000]   21:
[LR5]   22:   Var regs: Register;
[X42]   23:       buf : Byte;
[000]   24:
[BG1]   25:   Begin
[8R2]   26:     entry:=' ';
[FB3]   27:     With regs Do
[BG1]   28:       Begin
[NM1]   29:         AX:=$0800;
[W53]   30:         MsDos(regs);
[OY2]   31:         buf:=Lo(AX);
[FS3]   32:         If buf=0 Then
[BG1]   33:           Begin
[W53]   34:             MsDos(regs);
[OY2]   35:             buf:=Lo(AX);
[TTO]   36:           End;
[103]   37:         Case buf Of
[YK4]   38:           $31      : entry:='1';
[4L4]   39:           $32      : entry:='2';
[AL4]   40:           $33      : entry:='3';
[GL4]   41:           $34      : entry:='4';
[PL4]   42:           $35      : entry:='5';
[WL4]   43:           $36      : entry:='6';
[2M4]   44:           $37      : entry:='7';
[8M4]   45:           $38      : entry:='8';
[EM4]   46:           $39      : entry:='9';
[948]   47:           $0D      : entry:='E';   { Enter
        }
[9V6]   48:           $3B,$54 : entry:='F';   { F1
        }
[E88]   49:           $4B      : entry:='L';   { links
        }
[678]   50:           $4D      : entry:='R';   { rechts
        }
[R28]   51:           $1B      : entry:='X';   { Escape
        }
[TTO]   52:         End;
[TTO]   53:       End;
[TTO]   54:   End;
[000]   55:
[000]   56:
[665]   57: Procedure Beep;
[000]   58:
[BG1]   59:   Begin
[BE2]   60:     Sound(440);
[DD2]   61:     Delay(100);
[UV2]   62:     NoSound;
[TTO]   63:   End;
[000]   64:
[000]   65:
[N3P]   66: Procedure Tune(start,stop,inc:Integer);
                { spielt eine Tonfolge: }
[686]   67:
                { rauf oder runter      }
[FF5]   68:   Var step:Integer;
[000]   69:
[BG1]   70:   Begin
[CC3]   71:     step:=start;
[28D]   72:     While ((step<>stop) And (step>0) And (s
           tep<1000)) Do
[BG1]   73:       Begin
[X73]   74:         Sound(step);
[DD2]   75:         Delay(100);
[YV4]   76:         step:=step+inc;
[UY2]   77:         Nosound;
[TTO]   78:       End;
[TTO]   79:     End;
[000]   80:
[000]   81:
[K8M]   82: Procedure Line;
          { schreibt eine horizontale Linie }
[000]   83:
[BG1]   84:   Begin
[767]   85:     For n:=1 To 80 Do Write('-');
[TTO]   86:   End;
[000]   87:
[V75]   88: Procedure Help;
[000]   89:
[RY3]   90:   Var key:Char;
[000]   91:
[XE5]   92:   Procedure Header;
[BG1]   93:     Begin
[NF2]   94:       ClrScr;
[6M6]   95:       TextColor(LightGray);
[D76]   96:       TextBackground(Blue);
[U7B]   97:       GotoXY(1,2); Writeln('  H i l f s  -
           S e i t e  ');
[FP6]   98:       TextBackground(Black);
[OA4]   99:       GotoXY(1,4); Line;
[TTO]  100:     End;
[000]  101:
[BG1]  102:   Begin
[E72]  103:     key:='p';
[BU5]  104:     While key In ['p','P'] Do
[BG1]  105:     Begin
[7G2]  106:       Header;
[2T2]  107:       GotoXY(1,6);
[RWM]  108:       Write('Der Computer hat sich eine Kom
           bination aus vier Farben ausge');
[9BN]  109:       Write('dacht. Versuchen    Sie, diese
           mit möglichst wenigen Zügen zu ');
[D16]  110:       Writeln('finden.');
[XCL]  111:       Writeln('Sie bekommen allerdings eine
           kleine Hilfestellung:');
[9Y2]  112:       Writeln;
[05M]  113:       Write('Für jede richtige Farbe, die S
           ie an die richtige Stelle ge');
[82M]  114:       Write('setzt haben, bekommen Sie eine
           braune Markierung und für ');
[1VL]  115:       Write('jede richtige Farbe an der fal
           schen Stelle   eine Weiße. ');
[FZN]  116:       Writeln('Mit der Zeit können Sie also
           auf die Kombination schließen.');
[9Y2]  117:       Writeln;
[HUH]  118:       Writeln('Geben Sie Ihre Züge mit folg
           enden Tasten ein:');
[9Y2]  119:       Writeln;
[6L3]  120:       For n:=1 To 6 Do
[BG1]  121:         Begin
[AS3]  122:           TextColor(n);
[AB2]  123:           Write('ω');
[6M6]  124:           TextColor(LightGray);
[694]  125:           Writeln('  -  ',n);
[TTO]  126:         End;
[5A3]  127:       GotoXY(40,16);
[B1M]  128:       TextColor(Yellow);Write('ω');TextColo
           r(LightGray);Writeln('  -  7');
[9A3]  129:       GotoXY(40,17);
[1SL]  130:       TextColor(White);Write('ω');TextColor
           (LightGray);Writeln('  -  8');
[2U9]  131:       GotoXY(40,18); Writeln('0  -  9   (Lo
           ch)');
[M64]  132:       GotoXY(1,23); Line;
[CL6]  133:       TextColor(LightCyan);
[HM8]  134:       Write('Bitte weiter mit "C"...');
[YN1]  135:       key:=' ';
[BEB]  136:       Repeat Read(Kbd,key) Until key In ['c
           ','C'];
[6M6]  137:       TextColor(LightGray);
[7G2]  138:       Header;
[4T2]  139:       GotoXY(1,7);
[GLL]  140:       Write('Benutzen Sie die Pfeile der Ta
           statur, um den Kursor zu ');
[MEF]  141:       Writeln('bewegen und ENTER, um denZug
           zu beenden.');
[2V5]  142:       Writeln; Writeln;
[M6N]  143:       Writeln('Sie können zwischen drei Sch
           wierigkeitsgraden wählen:');
[9Y2]  144:       Writeln;
[ZUL]  145:       Writeln('1  -  Die Farben der Kombina
           tion kommen nur einzeln vor.');
[9Y2]  146:       Writeln;
[XDH]  147:       Writeln('2  -  Es kann mehrfach diese
           lbe Farbe vorkommen.');
[9Y2]  148:       Writeln;
[Y9N]  149:       Write('3  -  Die Kombination kann auc
```

```
          h Freistellen (Löcher) enthalten.');
[VP8]  150:       Writeln; Writeln; Writeln;
[XE6]  151:       Writeln('Viel Spaß!');
[M64]  152:       GotoXY(1,23); Line;
[CL6]  153:       TextColor(LightCyan);
[NXE]  154:       Write('Bitte weiter mit "C" oder zurü
          ck mit "P"...');
[YN1]  155:       key:=' ';
[RYC]  156:       Repeat Read(Kbd,key) Until key In ['c
          ','C','p','P'];
[291]  157:       Beep;
[TTO]  158:     End;
[TTO]  159:   End;
[000]  160:
[000]  161:
[O2R]  162: Procedure Combination;                { eine
          Farbkombination wird in 'comb' ge= }
[M57]  163:                                       { speic
          hert                                  }
[BG1]  164:   Begin
[MP5]  165:     comb[1]:=Random(27);
[NHF]  166:     If difficulty=3 Then For n:=1 To 4 Do c
          omb[n]:=Random(9)+1;
[7HF]  167:     If difficulty=2 Then For n:=1 To 4 Do c
          omb[n]:=Random(8)+1;
[EY5]  168:     If difficulty=1 Then
[BG1]  169:       Begin
[3Y5]  170:         comb[1]:=Random(8)+1;
[AVE]  171:         Repeat comb[2]:=Random(8)+1 Until c
          omb[1]<>comb[2];
[DKF]  172:         Repeat comb[3]:=Random(8)+1 Until (
          (comb[3]<>comb[1]) And
[XF5]  173:                                          (
          comb[3]<>comb[2]));
[MKF]  174:         Repeat comb[4]:=Random(8)+1 Until (
          (comb[4]<>comb[1]) And
[ZL6]  175:                                          (
          comb[4]<>comb[2]) And
[4G5]  176:                                          (
          comb[4]<>comb[3]));
[TTO]  177:       End;
[TTO]  178:   End;
[000]  179:
[000]  180:
[456]  181: Procedure DrawBoard;
[000]  182:
[BG1]  183:   Begin
[NF2]  184:     ClrScr;
[OR5]  185:     TextColor(Yellow);
[D76]  186:     TextBackground(Blue);
[6EA]  187:     GotoXY(18,2); Writeln('   S U P E R - M
          I N D   ');
[6H5]  188:     TextColor(Green);
[FP6]  189:     TextBackground(Black);
[37S]  190:     GotoXY(18,1); Writeln('▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄
          ▄▄▄▄▄▄▄▄');
[5FS]  191:     GotoXY(18,3); Writeln('▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄
          ▄▄▄▄▄▄▄▄');
[M7P]  192:     GotoXY(18,4); Writeln('┌─────┬─────┬──
          ──┬─────┐');
[NL3]  193:     For n:=1 To 9 Do
[BG1]  194:       Begin
[TVC]  195:         GotoXY(18,3+2*n); Writeln('|     |
          |     |     |');
[GLS]  196:         GotoXY(18,4+2*n); Writeln('├─────┼─
          ────┼─────┼─────┤');
[TTO]  197:       End;
[SHB]  198:     GotoXY(18,23); Writeln('|     |     |
          |     |');
[HPP]  199:     GotoXY(18,24); Writeln('└─────┴─────┴──
          ──┴─────┘');
[BWG]  200:     GotoXY(47,4); Writeln('┌───┬───┬───┬──
          ┐');
[NL3]  201:     For n:=1 To 9 Do
[BG1]  202:       Begin
[CDB]  203:         GotoXY(47,3+2*n); Writeln('|   |
          |   |   |');
[V7L]  204:         GotoXY(47,4+2*n); Writeln('├───┼──
          ┼───┼───┤');
[TTO]  205:       End;
[B3A]  206:     GotoXY(47,23); Writeln('|   |   |   |
          |');
[CBH]  207:     GotoXY(47,24); Writeln('└───┴───┴───┴──
          ┘');
[6BA]  208:     GotoXY(49,2); Writeln('von Jens Kock');
[6L3]  209:     For n:=1 To 6 Do
[BG1]  210:       Begin
[314]  211:         GotoXY(70,3+2*n);
[277]  212:         TextColor(n); Write('ω');
[92B]  213:         TextColor(LightGray); Writeln(' - '
          ,n);
[TTO]  214:       End;
```

```
[9A3]  215:     GotoXY(70,17);
[XL9]  216:     TextColor(Yellow); Write('ω');
[VEA]  217:     TextColor(LightGray); Writeln(' - 7');
[HA3]  218:     GotoXY(70,19);
[H99]  219:     TextColor(White); Write('ω');
[YEA]  220:     TextColor(LightGray); Writeln(' - 8');
[C67]  221:     GotoXY(70,21); Writeln('0 - 9');
[G56]  222:     TextColor(LightRed);
[XP8]  223:     GotoXY(1,7); Write('Beststand:');
[3R7]  224:     GotoXY(1,10); Write('Punkte:');
[6M6]  225:     TextColor(LightGray);
[BBA]  226:     GotoXY(1,23);Writeln('ESC - aufgeben');
[TTO]  227:   End;
[000]  228:
[000]  229:
[LFN]  230: Procedure Evaluate;                   { Der
          Zug des Spielers wird bewertet.    }
[R9B]  231:                                       { Für
          jeden richtigen Stein auf der      }
[5SN]  232:   Var d,e,temp,place:Integer;         { ric
          htigen Stelle: marks[place]:=1      }
[ZNL]  233:       c,m:Array [1..4] Of Integer;    { und
          für jeden richtigen Stein an       }
[3HB]  234:                                       { der
          falschen Stelle: marks[place]:=2.  }
[83H]  235:   Begin                               { 'so
          lution' wird auf True gesetzt.     }
[6UM]  236:     solution:=False;                  { wen
          n die richtige Kombination gefun=  }
[UUC]  237:     For d:=1 To 4 Do                  { den
          wurde.                             }
[BG1]  238:       Begin
[YB3]  239:         marks[d]:=0;
[304]  240:         c[d]:=comb[d];
[234]  241:         m[d]:=move[d];
[TTO]  242:       End;
[K92]  243:     place:=1;
[AH3]  244:     For d:=1 To 4 Do
[AV4]  245:       If m[d]=c[d] Then
[BG1]  246:         Begin
[LU4]  247:           marks[place]:=1;
[W34]  248:           place:=place+1;
[9Y1]  249:           m[d]:=-1;
[ZX1]  250:           c[d]:=-2;
[TTO]  251:         End;
[UGK]  252:     If ((marks[1]=1) And (marks[2]=1) And (
          marks[3]=1) And (marks[4]=1)) Then
[BG1]  253:       Begin
[8F5]  254:         solution:=True;
[BZ2]  255:         game:=False;
[DKO]  256:       End
[YH2]  257:     Else Begin
[AH3]  258:       For d:=1 To 4 Do
[FH3]  259:         For e:=1 To 4 Do
[P48]  260:           If ((d<>e) And (m[d]=c[e])) Then
[BG1]  261:             Begin
[UU4]  262:               marks[place]:=2;
[W34]  263:               place:=place+1;
[9Y1]  264:               m[d]:=-1;
[2Y1]  265:               c[e]:=-2;
[TTO]  266:             End;
[SEK]  267:       For d:=1 To 4 Do              { Hier w
          ird die Bewertung etwas vermischt. }
[7ZH]  268:         Begin                         { sodaß
          der Spieler die Kombination nicht  }
[6VH]  269:           e:=Random(4)+1;             { aus de
          r Anordnung der Bewertung er=      }
[3TE]  270:           temp:=marks[d];             { schlie
          ßen kann.                          }
[Y36]  271:           marks[d]:=marks[e];
[RA5]  272:           marks[e]:=temp;
[TTO]  273:         End;
[CKC]  274:       e:=1;
                  { Bewertung ausgeben }
[AH3]  275:       For d:=1 To 4 Do
[BG1]  276:         Begin
[L35]  277:           GotoXY(45+4*e,27-2*y);
[RS4]  278:           If marks[d]=1 Then
[BG1]  279:             Begin
[NN5]  280:               TextColor(Brown);
[AB2]  281:               Write('ω');
[YN1]  282:               e:=e+1;
[TTO]  283:             End;
[WS4]  284:           If marks[d]=2 Then
[BG1]  285:             Begin
[XK5]  286:               TextColor(White);
[AB2]  287:               Write('ω');
[YN1]  288:               e:=e+1;
[TTO]  289:             End;
[TTO]  290:         End;
[TTO]  291:     End;
```

```
[TTO]  292:   End;
[000]  293:
[000]  294:
[WGS]  295: Procedure WriteColour(stone:Integer);    {
       Ausgaberoutine für Zug und Kombi=  }
[UM6]  296:                                          {
       nation                                 }
[BG1]  297:   Begin
[CND]  298:     If ((stone>=1) And (stone<=6)) Then Tex
       tColor(stone);
[4YA]  299:     If stone=7 Then TextColor(Yellow);
[3UA]  300:     If stone=8 Then TextColor(White);
[875]  301:     If stone=9 Then
[BG1]  302:       Begin
[6M6]  303:         TextColor(LightGray);
[AB2]  304:         Write('O');
[DKO]  305:       End
[9P4]  306:     Else Write('ø');
[TTO]  307:   End;
[000]  308:
[000]  309:
[1AE]  310: Begin
                                { Hauptprogramm }
[N83]  311:   hiscore:=7;
[X42]  312:   Repeat
[9C2]  313:     score:=0;
[FWO]  314:     y:=1;
[ES2]  315:     entry:='F';
[CRM]  316:     While entry='F' Do          { Wenn die H
       ilfs-Seite verlassen wird, muß das }
[VSF]  317:     Begin                       { Menü neu a
       usgegeben werden.                      }
[AE5]  318:       TextColor(Black);
[NL2]  319:       Clrscr;
[OR5]  320:       TextColor(Yellow);
[D76]  321:       TextBackground(Blue);
[PVA]  322:       GotoXY (27,2); Writeln('   S U P E R
       - M I N D   ');
[HT5]  323:       TextBackground(Red);
[OY6]  324:       GotoXY(32,12); Writeln(' 1 ');
[AY6]  325:       GotoXY(32,14); Writeln(' 2 ');
[MY6]  326:       GotoXY(32,16); Writeln(' 3 ');
[FV6]  327:       Textbackground(Black);
[6H5]  328:       TextColor(Green);
[7BA]  329:       GotoXY(34,4); Writeln('von Jens Kock'
       );
[LOB]  330:       GotoXY(32,9); Writeln('Wie spielen Si
       e?');
[6H9]  331:       GotoXY(36,12); Writeln('- mäßig');
[2Z7]  332:       GotoXY(36,14); Writeln('- gut');
[Z5A]  333:       GotoXY(36,16); Writeln('- exzellent')
       ;
[M64]  334:       GotoXY(1,23); Line;
[XK5]  335:       TextColor(White);
[Z28]  336:       GotoXY(2,24); Write('F1  -  Hilfe');
[G18]  337:       GotoXY(68,24);Write('ESC  -  Ende');
[TUE]  338:       Repeat ScanKeyboard Until entry In ['
       1','2','3','F','X'];
[291]  339:       Beep;
[1Z6]  340:       If entry='F' Then Help;
[KS4]  341:       If entry='X' Then
[BG1]  342:         Begin
[WKB]  343:           GotoXY(30,24);Write('Sind Sie sic
       her (j/n)?');
[8R2]  344:           entry:=' ';
[KKE]  345:           Repeat Read(Kbd,entry) Until entr
       y In ['j','J','n','N'];
[8Z5]  346:           entry:=UpCase(entry);
[8R4]  347:           If entry='J' Then
[BG1]  348:             Begin
[NF2]  349:               ClrScr;
[6M6]  350:               TextColor(LightGray);
[5A1]  351:               Halt;
[DKO]  352:             End
[844]  353:           Else entry:='F';
[TTO]  354:         End;
[TTO]  355:     End;
[LUH]  356:     ClrScr;                             { S
       pielanfang: Kombination ausdenken, }
[USL]  357:     Val(entry,difficulty,code);         { S
       pielfeld zeichnen, usw.                }
[NN3]  358:     Combination;
[CW2]  359:     DrawBoard;
[RH8]  360:     GotoXY(13,7); Writeln(hiscore);
[GP2]  361:     game:=True;
[6MH]  362:     While game=True Do
                  { Anfang der Spiel-Schleife }
[BG1]  363:     Begin
[WSA]  364:       For count:=1 To 4 Do move[count]:=-
       1;
[EWO]  365:       x:=1;
[6M6]  366:       TextColor(LightGray);
[L84]  367:       score:=score+1;
[6L7]  368:       GotoXY(13,10); Writeln(score);
[SRE]  369:       Repeat
                  { Anfang der Zug-Schleife }
[S65]  370:         GotoXY(15+6*x,25-2*y);
[U5G]  371:         Repeat ScanKeyboard Until entry I
       n ['1','2','3','4','5','6','7','8',
[3N3]  372:
            '9','L','R','E','X'];
[21A]  373:         If entry='R' Then If x<4 Then x:=
       x+1;
[41A]  374:         If entry='L' Then If x>1 Then x:=
       x-1;
[8S9]  375:         If ((entry>='1') And (entry<='9')
       ) Then
[BG1]  376:           Begin
[XN7]  377:             Val(entry,move[x],code);
[BK6]  378:             WriteColour(move[x]);
[MO7]  379:             If x<4 Then x:=x+1 Else x:=1;
[TTO]  380:           End;
[9XH]  381:       Until ((entry='E') Or (entry='X'));
                    { Ende der Zug-Schleife }
[291]  382:       Beep;
[2T1]  383:       y:=y+1;
[PT6]  384:       If y>10 Then game:=False;
[KS4]  385:       If entry='X' Then
[BG1]  386:         Begin
[7P5]  387:           solution:=False;
[BZ2]  388:           game:=False;
[DKO]  389:         End
[694]  390:       Else Evaluate;
[42E]  391:     End;
                  { Ende der Spiel-Schleife }
[PT8]  392:     GotoXY(18,2); Writeln('
                ');
[LT8]  393:     GotoXY(18,1); Writeln('
                ');
[TT8]  394:     GotoXY(18,3); Writeln('
                ');
[GTE]  395:     For n:=1 To 4 Do
                    { Ergebnisse ausgeben }
[BG1]  396:       Begin
[ZX3]  397:         GotoXY(15+6*n,2);
[6E6]  398:         WriteColour(comb[n]);
[TTO]  399:       End;
[F85]  400:     If solution Then
[BG1]  401:       Begin
[W27]  402:         If score<hiscore Then
[BG1]  403:           Begin
[PF5]  404:             hiscore:=score;
[XK5]  405:             TextColor(White);
[FK7]  406:             GotoXY(13,7); Write(hiscore);
[HT5]  407:             TextBackground(Red);
[27A]  408:             GotoXY(1,13); Write('Beste Pu
       nktzahl!');
[583]  409:             Tune(440,700,20);
[FP6]  410:             TextBackground(Black);
[DKO]  411:           End
[AO1]  412:         Else
[BG1]  413:           Begin
[VY7]  414:             TextColor(LightMagenta);
[CE6]  415:             GotoXY(1,13); Write('Gut!');
[Z63]  416:             Tune(400,500,10);
[TTO]  417:           End;
[DKO]  418:       End
[AO1]  419:     Else
[BG1]  420:       Begin
[NO3]  421:         GotoXY(1,13);
[6H5]  422:         TextColor(Green);
[X93]  423:         n:=Random(5);
[C32]  424:         Case n Of
[LW4]  425:           0 : Write('Pech!');
[OA5]  426:           1 : Write('Schade!');
[TA6]  427:           2 : Write('Verloren!');
[H55]  428:           3 : Write('Schon?');
[EL7]  429:           4 : Write('Macht nichts!')
[TTO]  430:         End;
[7A3]  431:         Tune(700,300,-20);
[TTO]  432:       End;
[AE5]  433:     TextColor(Black);
[6M6]  434:     TextColor(LightGray);
[U66]  435:     GotoXY(1,23); Write('          ');
[FPK]  436:     GotoXY(1,21); Write('Bitte mit ESC');
                  { zurück zum Spielanfang }
[SY9]  437:     GotoXY(1,23); Write('weitermachen  ')
       ;
[T7B]  438:     Repeat ScanKeyboard Until entry='X';
[C82]  439:   Until 1=2;
[3DF]  440: End.
                { Ende des Hauptprogrammes }
```

Listing. Der Quellcode zu »Super-Mind« erstellt mit Turbo Pascal 3.0

SForth 3.1:
Ein Compiler im Selbstbau

In den Ausgaben 9'87 bis 11'87 der DOS International wurde der SForth-Compiler als Software-Projekt vorgestellt. Der Compiler erzeugt sehr schnellen Maschinencode und ist im Befehlsumfang an die Programmiersprache Forth angelehnt. Mittlerweile ist durch intensive Mitarbeit der DOS-Leser eine völlig überarbeitete und stark erweiterte Version 3.1 dieses Compilers verfügbar.

Reiner Maschinencode ist schnell – das weiß jeder Computerbesitzer. Genauso bekannt ist aber auch, daß Assembler oder gar echter Maschinencode für den Menschen nur sehr umständlich und schwierig zu handhaben ist. Höhere Programmiersprachen wie Basic oder Pascal sind einfacher zu benutzen, bieten aber weder die Geschwindigkeit noch die Effizienz maschinennaher Sprachen. Programmiersprachen wie Forth sind hier ein guter Kompromiß: Sie arbeiten einerseits sehr maschinennah und sind daher sehr schnell, bieten aber gleichzeitig einen ähnlichen Bedienungskomfort wie ein Compiler für eine echte Hochsprache. Mit dem hier vorgestellten SForth-Compiler sind schnelle und maschinennahe Programme kein Problem mehr.

Die Wahl einer Forth-ähnlichen Sprache für diesen Compiler hat verschiedene Gründe:

– Forth ist schnell. Forth-Programme laufen mit einer Geschwindigkeit, die in vielen Fällen nur unwesentlich unter der von reinem Assemblercode liegt.

– Forth ist kompakt. Durch die spezielle Struktur dieser Sprache ist der Forth-Code sehr effizient und erlaubt sehr komplexe Programme mit extrem wenig Speicherbedarf.

– Forth ist leicht. Entgegen landläufiger Meinung ist Forth im Grunde eine sehr einfache Sprache. Lassen Sie sich nicht durch das etwas merkwürdige Aussehen von Forth-Programmen täuschen – Forth ist einfach zu erlernen und noch einfacher einzusetzen.

– Forth ist erweiterbar. Eine der wesentlichsten Eigenschaften von Forth ist die in der Sprache enthaltene Fähigkeit zur Selbsterweiterung. Wenn Sie irgendwelche speziellen Funktionen brauchen, dann schreiben Sie sie einfach hin und übernehmen sie ohne viel Aufwand in den Sprachumfang. Es gibt keinen Unterschied zwischen vordefinierten und selbstgeschriebenen Befehlen.

– Forth ist strukturiert. Programmieren mit Forth heißt: Kleine Bausteine zu größeren zusammensetzen, diese wieder zu noch größeren und so fort. Es gibt keinen Sprungbefehl, weil er

einfach nicht notwendig ist. Haben Sie eine Funktion einmal programmiert, dann brauchen Sie sich anschließend nie wieder mit ihrem internen Aufbau zu beschäftigen. Sie setzen sie einfach nur noch wie eine »Black Box« ein, um das eigentliche Problem zu lösen.

– Forth ist portabel. Es gibt praktisch für jeden Computertyp vom Heimcomputer bis zum Großrechner verschiedene Forth-Compiler, die aber alle gewissen Standards gehorchen. Ein halbwegs geübter Forth-Programmierer hat keine Probleme beim Umstieg auf einen anderen Computer.

Zu allen diesen Vorzügen kommt schließlich noch einer, der den Ausschlag für die Wahl einer Forth-ähnlichen Sprache als Zielsprache für den Compiler gab: Aufgrund der sehr einfachen Sprachstruktur von Forth und der Erweiterbarkeit dieser Sprache ist es vergleichsweise einfach, einen entsprechenden Compiler zu schreiben. Im Falle von Forth muß ein solcher Compiler nur sehr wenige Grundmuster der Sprache kennen und übersetzen können – den ganzen Rest kann man dann getrost in Forth selbst schreiben. Wem die Vorstellung, eine Programmiersprache quasi in sich selbst schreiben zu können, gewisse Probleme bereitet, der hat auf den folgenden Seiten Gelegenheit, sich von der Tragfähigkeit dieses Konzeptes zu überzeugen.

SForth erzeugt im Gegensatz zu Standard-Forth echten Maschinencode

Doch zunächst ein kleiner Ausflug zu den Grundlagen von Forth-Compilern im Allgemeinen. Forth wird üblicherweise gar nicht als echter Compiler implementiert, sondern stellt ein Zwischending zwischen Interpreter und Compiler dar. In einem echten Forth-System ist Forth gleichzeitig Programmiersprache und Betriebssystem-Umgebung. Der Interpreter-Teil von Forth führt Benutzer-Eingaben und -Befehle aus und ermöglicht das Arbeiten mit virtuellem Speicher. Ähnlich wie bei einem Basic-Interpreter bemerkt der Anwender auch hier das eigentliche Betriebssystem (zum Beispiel MS-DOS) gar nicht, sondern arbeitet ausschließlich im Forth-System.

Neue Befehle (die in Forth »Worte« heißen), werden im Forth-System entwickelt, wobei der Interpreter bei Bedarf automatisch den Forth-Compiler startet. Der Forth-Compiler erzeugt aber keinen reinen Maschinencode, wie es etwa ein C- oder ein Pascal-Compiler tut, sondern einen sogenannten »gefädelten« Code. Dabei sind die einzelnen Forth-Worte innerhalb eines sogenannten »Vocabulary« abgelegt und über Zeigerstrukturen miteinander verflochten. Ebenfalls mit abgelegt werden die Namen der Worte und Informationen über die Art des Wortes (Maschinencode-Wort, Konstante, Variable, High-Level-Wort und so weiter).

Bei der Programmausführung wird dieser gefädelte Code von einem speziellen Programmteil aus interpretiert und ausgeführt. Aufgrund der effizienten Struktur dieses Codes erfolgt die Ausführung in der Regel nur wenig langsamer als bei echtem Maschinencode. Doch der hier vorgestellte SForth-Compiler beseitigt auch diese Einschränkung: Er erzeugt reinen Maschinencode, der direkt und ohne Umwege vom Prozessor ausgeführt wird. In Anlehnung an das ursprüngliche Forth-Konzept legt aber auch SForth Informationen wie Typ und Name von Worten unmittelbar im erzeugten Code ab, allerdings ohne die Ausführungsgeschwindigkeit zu beeinträchtigen. Wie Sie im folgenden sehen, erleichtert dies die Arbeit des Compilers ganz enorm. Bild 1 zeigt schematisch den Aufbau des vom SForth-Compiler erzeugten Codes.

Symbol	Bedeutung
n	Integer-Zahlen (16 Bit)
c	Textzeichen oder Byte-Zahlen (8 Bit)
d	doppeltgenaue Zahl (32 Bit)
u	vorzeichenlose Zahl (16 Bit)
addr	Speicheradresse (16 Bit)

Tabelle 1. Häufige Symbole in Stack-Diagrammen

Die SForth-Story: Von der Sprachdefinition zum Compiler-Entwurf

SForth wurde entworfen als einfacher und übersichtlicher Compiler, der eine vergleichsweise kleine Grundmenge an Sprachstrukturen übersetzen sollte. Die Erweiterbarkeit der Sprache erlaubte es dann, ein leistungsfähiges System aus einfachsten Anfängen hochzuziehen. Ausgehend von diesem Grundkonzept entstand ein Entwurf der zu übersetzenden Sprache, an der der Compiler sich orientierte.

Der Entwurf sieht fünf grundlegende Worttypen vor: Konstanten, Variable, Makros, Prozeduren und System-Worte. Während sich die ersten vier Typen in SForth selbst erzeugen lassen, ist dies (im Gegensatz zu Standard-Forth) bei System-Worten nicht der Fall. System-Worte sind gerade die (wenigen) Worte, die man auf der Ebene eines von einem Compiler zu übersetzenden Forth nicht mit überschaubaren Aufwand als eigentliche Forth-Worte implementieren kann. Dazu gehören die Definitionsworte, die beispielsweise die Variablen- oder Konstanten-Definition einleiten sowie natürlich die Struktur-Worte wie »if«, »endif«, »do« oder »loop«, um nur einige Beispiele zu nennen. Diese System-Worte sind die einzigen Worte, die der Compiler von vornherein kennt. Alles andere »lernt« er erst während des Übersetzens.

SForth ist keine Eintagsfliege! Der Compiler wurde erdacht, um schnelle Systemprogrammierung mit den Vorteilen einer strukturierten Sprache zu kombinieren. Sie schreiben mit SForth Programme, die praktisch genauso schnell und kompakt sind wie echter Assembler, die aber viel einfacher und komfortabler zu entwerfen, ändern und weiterzuentwickeln sind. Ihre Anregungen, Fragen oder mit SForth entwickelten Utilities und Programme sind uns dabei willkommen.

SForth ist kein statisches Programm, sondern ein ständig in Wachstum und Überarbeitung befindliches Entwicklungssystem. Um das ganze Projekt in geordneten Bahnen laufen zu lassen, bitten wir alle an SForth interessierten Leser, sämtliche Verbesserungsvorschläge, Compiler-Erweiterungen oder selbstgeschriebene Vocabularies an die DOS-Redaktion zu schicken. Alle Zusendungen werden überprüft und bei Eignung als offizielle Compiler-Erweiterungen veröffentlicht und allen PC-Anwendern zugänglich gemacht. Bitte arbeiten Sie auf keinen Fall mit »wilden« Compiler-Erweiterungen (außer für den rein privaten Gebrauch), um eine möglichst weitgehende Kompatibilität aller SForth-Programme sicherzustellen.

Natürlich wird jeder veröffentlichte Beitrag rund um SForth entsprechend honoriert. Schreiben Sie an die Redaktion DOS International, Stichwort SForth, Elfenstr. 40, 8000 München 83

SForth arbeitet wie jeder Forth-Compiler mit der »umgekehrten polnischen Notation«

Um den Übersetzungsvorgang zu verstehen, muß man zunächst etwas mehr über die Sprache selbst wissen: Forth (und natürlich auch SForth) arbeitet mit einem Daten- und einem Return-Stack. Der Return-Stack dient zum Speichern der Rückkehradressen bei Prozeduraufrufen und zum temporären Zwischenspeichern von Daten, der Datenstack dagegen zur Manipulation der aktuellen Daten. Alle Rechenoperationen laufen bei Forth über diesen Datenstack. Dabei wird die sogenante »umgekehrte polnische Notation« verwendet, die unter dem Kürzel »UPN« jedem Benutzer eines HP-Taschenrechners im

Prinzip bekannt ist. Bei dieser Schreibweise werden stets zuerst die Daten genannt, mit denen etwas geschehen soll, und dann erst die Operation, die mit den Daten ausgeführt werden soll.

Aus der Rechnung »3 + 4 =« wird in UPN-Schreibweise also: »3 4 +«. Das Gleichheitszeichen wird nicht benötigt, denn durch »+« wird die Addition sofort ausgeführt. Bild 2 zeigt die Verhältnisse auf dem Datenstack während der Ausführung dieser Rechnung.

Neue Worte (Prozeduren) werden in Forth durch die sogenannte »Colon-Definition« erzeugt. Der Name kommt von dem Doppelpunkt (englisch Colon), der jede solche Definition einleitet. Hinter dem Doppelpunkt steht der Name der neuen Prozedur, gefolgt von den Forth-Worten, die bei Aufruf der neuen Prozedur ausgeführt werden sollen. Beendet wird die Colon-Definition durch ein Semikolon. Ein Beispiel für eine solche Colon-Definition ist:

```
: EinsPlus 1 + ;
```

Diese Definition legt ein neues Wort »EinsPlus« fest, das die Zahl Eins auf den Stack legt und anschließend eine Addition durchführt. Es wird also im Endeffekt die vor dem Aufruf von »EinsPlus« auf dem Stack liegende Zahl um Eins erhöht. Operatoren wie »+«, die zwei Werte miteinander verknüpfen, arbeiten in Forth immer mit den beiden obersten Elementen des Datenstacks, dem TOS (Top of Stack) und dem sogenannten »Second« (der zweite Stackeintrag). Beachten Sie, daß nach dem Doppelpunkt und vor dem Semikolon stets ein Leerzeichen vorhanden sein muß. Im Gegensatz zu anderen Sprachen sind diese Zeichen bei Forth ganz normale Worte und keine Interpunktionszeichen.

Der Compiler hält zwei Worte nur dadurch auseinander, daß mindestens ein Leerzeichen zwischen beiden steht. Ansonsten dürfen beliebige Zeichenfolgen, auch Sonderzeichen, als Namen für Worte verwendet werden.

Bei der Übersetzung einer Colon-Definition legt Standard-Forth eine Liste der in der Prozedur enthaltenen anderen Worte an und geht diese Liste bei Aufruf des neuen Wortes durch. SForth dagegen compiliert reinen Maschinencode. Daher werden die einzelnen Worte je nach Ihrem Typ unterschiedlich behandelt. Im Normalfall werden die einzelnen Worte direkt in Call-Befehle des 8088/8086-Prozessors übersetzt.

Als Erweiterung zum Standard-Forth erlaubt SForth jedoch auch die Definition von Makros. Makros sind formal genauso wie Colon-Definitionen aufgebaut, beginnen aber mit dem Schlüsselwort »makro« anstelle des Doppelpunktes. Bei der Definition eines Makros wird der Name genau wie bei der Colon-Definition auch in das Dictionary, also in den laufenden Programmcode, eingebunden. Auch das Übersetzen der einzelnen Befehle des Makros geht analog vonstatten. Allerdings wird bei einem Makro kein Ret-Befehl als Abschluß compiliert, wie das bei der Colon-Definition der Fall ist. Der eigentliche Unterschied liegt aber in der Art des Aufrufs: Wenn in einer anderen Definition ein Wort vom Prozedur-Typ vorkommt, dann wird wie bereits gesagt ein Call-Befehl zum Aufruf der entsprechenden Prozedur compiliert. Ist das Wort aber vom Makro-Typ, dann kopiert der Compiler den kompletten Code des Makros. Somit sind Makros geeignet für kurze Worte, bei denen es auf Ausführungsgeschwindigkeit ankommt. Längere Makros sind in der Regel nicht sinnvoll. Ein Makro mit einer Codelänge von beispielsweise 20 Byte erzeugt ja bei jedem Aufruf diese 20 Byte zusätzlichen Code, eine entsprechende Prozedur dagegen immer nur die drei Byte für den Call-Befehl, egal wie lang die Prozedur selber ist.

Eine wichtige Einschränkung für den Gebrauch von Makros ist noch zu erwähnen: Da bei jedem Makroaufruf der komplette Code kopiert wird, muß dieser Code relokatibel, also frei im Speicher verschiebbar sein. Aus diesem Grunde sind in Makros keine Prozeduraufrufe erlaubt. Prozeduraufrufe arbeiten beim 8088/8086-Prozessor nämlich mit relativen Adressen, sind also abhängig von der Speicherstelle, an der der Aufruf steht. Ebenfalls innerhalb von Makros nicht erlaubt sind Kontrollstrukturen wie »if...endif« und Aufrufe vordefinierter Variablen- und Konstantenworte. Echte Konstanten sind jedoch erlaubt.

SForth kennt natürlich auch Variablen und Konstanten

SForth unterscheidet zwischen echten Konstanten und dem Worttyp »constant«. Eine echte Konstante kann entweder eine Zahl oder ein ASCII-Zeichen sein. Zahlen dürfen mit oder ohne Vorzeichen und mit einem führenden Dollar-Zeichen auch in hexadezimaler Schreibweise angegeben werden. ASCII-Zeichen müssen mit einem Apostroph beginnen und enden. Nur ein Zeichen ist jeweils zugelassen. Beispiele für gültige Konstanten sind:

```
-1, 0, 8088, $b800, 'c', '2'
```

Der Compiler erzeugt für echte Konstanten einen Code, um den Wert der Konstanten auf den Datenstack zu le-

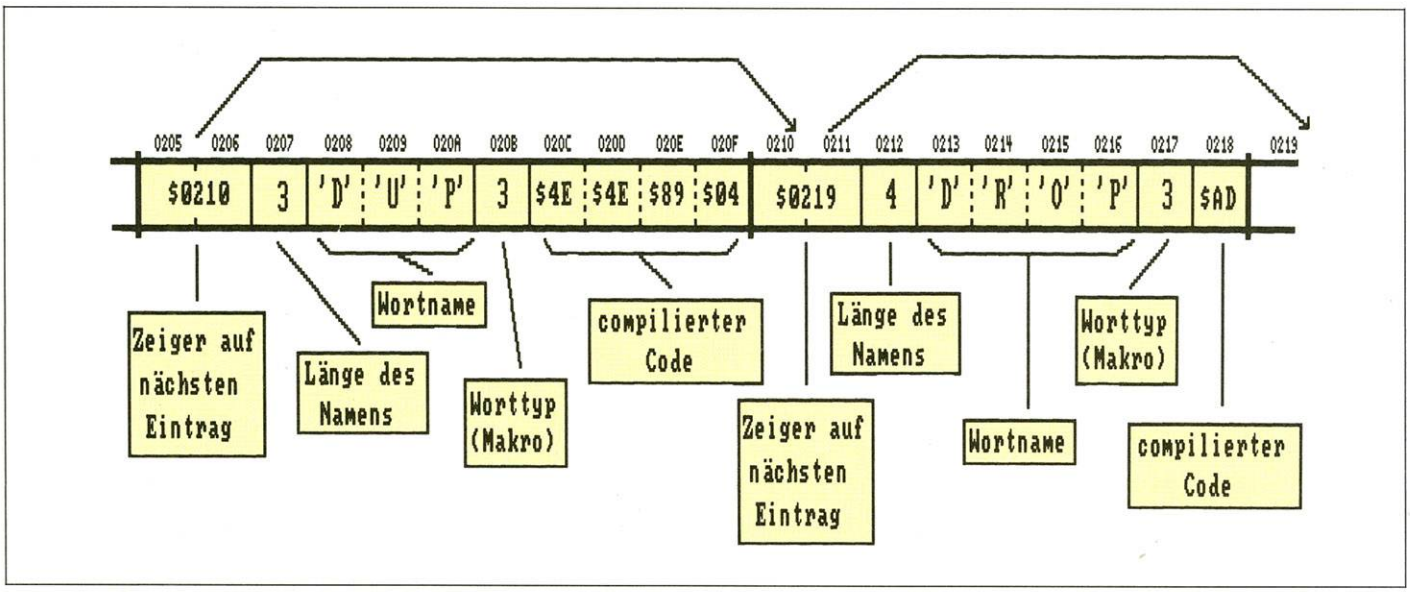


Bild 1. Aufbau des SForth-Codes

SForth Entwicklungsprotokoll

1986 erste Ansätze zur Programmierung auf Schneider CPC 464 unter CP/M 80.

Turbo Forth Versionen 0.x (April/Mai 1987)
Programmierung erster Versuchssysteme auf einem IBM-kompatiblen PC (Versionen 0.x). Konzeption der grundsätzlichen Arbeitsweise des Compilers.

Turbo Forth Version 1.0 (Juni 1987)
Turbo Forth Version 1.0 lauffähig. Diese Version des Compilers erzeugt reinen Maschinencode ohne Wort-Header. Symboltabellen werden als verkettete Liste temporär angelegt. Vorteil: Extrem kompakter Code. Nachteil: Vocabulary-Verwaltung recht schwierig, Compiler für Abdruck zu umfangreich.

Turbo Forth Version 2.0 (Juli 1987)
Programmierung von Turbo Forth 2.0. Diese Version entspricht bereits weitgehend der in der DOS Ausgabe 9'87 abgedruckten Version, weist jedoch eine andere Behandlung von Variablen und Konstanten auf. In der Version 2.0 werden bei Definition einer Variablen lediglich zwei Byte zur Aufnahme des Variablen-Wertes im Dictionary reserviert; Konstanten belegen bei Definition gar keinen Dictionary-Platz. Jeder Aufruf einer Variablen oder Konstanten erzeugt dagegen stets sieben Byte Code, um den entsprechenden Wert auf den Stack zu legen. Der Compiler selbst kann durch die gewählte spezielle Dictionary-Struktur vergleichsweise klein gehalten werden. Zu dieser Version des Compilers erfolgt die erste Codierung des Standard-Vocabulars »forth«.

Vocabulary Forth 1.0 (Juli 1987)
Diese erste Version des Forth-Vokabulars gehört zur Compiler-Version 2.0 und unterstützt den größten Teil des FIG-Forth-Wortschatzes, soweit für eine reine Compiler-Implementation sinnvoll. Diese Version des Forth-Vokabulars wurde auf der Databox-Diskette zur DOS Ausgabe 9'87 mit ausgeliefert und ist leider hinsichtlich der Worte »cmove« und »fill« inkompatibel zur abgedruckten Version 2.1 des Compilers (das Problem rührt von der geänderten Variablenbehandlung der Compiler-Version 2.1 her).

Turbo Forth Version 2.1 (Juli 1987)
In der DOS Ausgabe 9'87 ist die Version 2.1 des Turbo Forth-Compilers abgedruckt. Gleichzeitig erfolgt eine Namensänderung in »SForth«, um Namenskonflikte mit Borland-Software zu vermeiden. Wichtigster Unterschied zur Version 2.0 ist eine neuartige Behandlung von Variablen und Konstanten. In der Version 2.1 wird der Code, um den entsprechenden Wert auf den Stack zu legen, bereits bei der Definition einer Variablen oder Konstanten erzeugt; jeder Aufruf benötigt dagegen nur noch drei Byte für einen Call-Befehl. Nähere Informationen zur Version 2.1 siehe DOS Ausgabe 9'87.

Bekannte Fehler Turbo Forth 2.1:
(1) Bei Zeichenkonstanten werden Kleinbuchstaben nach Großbuchstaben konvertiert. Beispiel: Aus »'a'« macht der Compiler »'A'«.
(2) Programme ohne »main« werden vom Compiler nicht bemängelt. Das erzeugte Com-File ist in einem solchen Fall jedoch nicht lauffähig und stürzt beim Aufruf sofort ab.
(3) Alle neuen Worte, die mit dem Hochkomma-Zeichen »'« beginnen, versucht der Compiler als Char-Konstanten zu interpretieren. Falls dies nicht gelingt, erfolgt in jedem Falle eine Fehlermeldung. Damit ist das Hochkomma nur sehr eingeschränkt in neuen Namen zu verwenden.

Vocabulary Forth 1.1 (August 1987)
Diese Version wird in der Ausgabe 10'87 der DOS International veröffentlicht. Das Vocabulary Forth 1.1 gehört zur Compiler-Version 2.1 (nicht 2.0 !). Gegenüber der Version 1.0 sind einige Änderungen im Zusammenhang mit der seit der Compiler-Version 2.1 geänderten Variablen- und Konstanten-Handhabung geschehen. Geändert wurden in diesem Zusammenhang die Worte »cmove« und »fill«. Wegen der vergessenen Änderung für das Wort »dos« funktioniert dieses (ausgerechnet !) nicht richtig.

Bekannte Fehler Forth-Vocabulary 1.1:
(4) Die booleschen Operatoren »and«, »or« und »xor« erzeugen fehlerhaften Code
(5) »digit« erkennt bei Hex-Zahlen keine Kleinbuchstaben
(6) »(number)« erzeugt keine echten doppeltgenauen Zahlen
(7) Die Dezimalpunkt-Erkennung bei »number« ist fehlerhaft
(8) Der MS-DOS-Aufruf arbeitet fehlerhaft

SForth Version 2.2 (September 1987)
Die Fehler (1) und (2) sind behoben. Der Korrektur von (3) stehen noch grundsätzliche Kompatibilitätserwägungen (sowohl zur früheren SForth-Versionen als auch zu Standard-Forth) entgegen. Die Prozedur »dump« steht nun in verbesserter Form zur Verfügung und erzeugt eine übersichtlichere Auflistung des Dictionaries als zuvor (Erwin Schomburg). Alle Änderungen sind in der DOS Ausgabe 11'87 dokumentiert.

Bekannte Fehler SForth Version 2.2:
(9) Der Dictionary-Pointer »dp« wird mit einem konstanten Wert von 4000hex initialisiert. Dadurch ist ein Sicherheitsabstand zum eigentlichen SForth-Dictionary gewährleistet; bei sehr langen Programmen ist jedoch eine Überschneidung möglich. Wünschenswert wäre eine Initialisierung von »dp« auf das tatsächliche Ende des Dictionaries nach der Compilation. Zu beachten ist die unterschiedliche Bedeutung von »dp« bei Standard-Forth und SForth. SForth verwendet »dp« derzeit nur als Zeiger auf einen Laufzeit-Heap; zur Compile-Zeit hat »dp« keine Bedeutung.

Vocabulary Forth 1.2 (September 1987)
Die Fehler (4), (6) und (7) sind korrigiert (Erwin Schomburg, Klaus Scheller). Die Korrektur wird in der DOS Ausgabe 11'87 dokumentiert. Zu Fehler (5) wird ein Vorschlag für ein neues »digit« (Erwin Schomburg) in der gleichen Ausgabe veröffentlicht, aber noch nicht offizell in das Forth-Vocabulary aufgenommen. Es steht dazu die Überlegung an, ob man die Verwendung von hexadezimalen Zahlen nicht an den schließlich auch vom Compiler verwendeten Turbo Pascal-Standard anlehnen sollte (führendes Dollar-Zeichen). Fehler (8) hat sich durch Entfernen des Wortes »dos« aus dem Forth-Vocabulary erledigt.

Vocabulary System 1.0 (September 1987)
In der Ausgabe 11'87 der DOS International wird ein Vorschlag für ein Vocabulary zum Zugriff auf die Betriebssystemebene eines PCs veröffentlicht. Enthalten sind unter anderem die MS-DOS- und BIOS-Aufrufe, segmentierte Speicheradressierung und eine sehr einfache Stringbehandlung. Obwohl sicher ein Schritt in die richtige Richtung, überzeugt das System-Vocabulary in dieser Form bei näherer Betrachtung noch nicht ganz. Es enthält zu viele Umständlichkeiten und Inkonsequenzen, die allerdings zum Teil auf grundlegenden Konzepten der bisherigen Compiler- und Forth-Vocabulary-Versionen beruhen. Die Verwendung dieses Vocabularies wird in der vorliegenden Version nicht empfohlen, da eine tiefgreifende Umorganisation geplant ist.

Zwischenbilanz (Ende September 1987)
Folgende Compiler-Konzepte haben sich als nicht tragfähig erwiesen und werden daher geändert:
Die Verwendung der beiden Spezialvariablen »ss%« und »ds%« sollte die Speichersegmentierung beim PC softwareseitig auf einfache Art und Weise unterstützen. Der Gedanke war, bestimmte Befehle (»cmove«, »fill«, etc.) auf transparente Weise segmentübergreifend implementieren zu können. Das Vorbild war dabei der Befehl »defseg« von GW-Basic. Dieses Konzept bringt aber deutliche Nachteile mit sich, wenn neue System-Worte für weitere Vocabularies definiert werden, die mit Befehlen wie »cmove« oder »fill« arbeiten. Es ist dann ein sehr großer Aufwand nötig, um den korrekten Ablauf zu gewährleisten und unerwünschte Nebeneffekte zu vermeiden. »ds%« und »ss%« können ja sowohl von Systemworten wie auch von Anwenderworten geändert werden. Mithin darf ein Systemwort keine Annahmen über den Inhalt von »ds%« und »ss%« machen, beziehungsweise muß dafür Sorge tragen, daß die Variableninhalte gerettet und restauriert werden. Dies führt in der Praxis zu unnötigem, ständigem Programm-Overhead. Besser erscheint unter diesen Gesichtspunkten die Implementierung spezieller neuer Cmove- und Fill-Worte für Segmentüberschreitung.
Die Spezialvariablen »ax«, »bx«, »cx« und »dx« waren ursprünglich dazu vorgesehen, die Inhalte der wichtigsten Prozessorregister bei MS-DOS- und BIOS-Aufrufen zu übernehmen. Vorbild war hier der Register-Record, den Turbo Pascal bei MS-DOS-Aufrufen verwendet. Es zeigt sich jedoch, daß auch diese Methode beim SForth-Compiler zu unnötigem Programm-Overhead führt. Einfacher, überschaubarer und kürzer ist statt dessen die Codierung spezieller Worte wie beispielsweise »bx@« oder »cx!« zum direkten Ansprechen der Prozessorregister.

SForth Version 2.3 (September 1987)
Die Versorgung der Spezialvariablen »ss%« und »ds%« mit Initialisierungswerten entfällt wegen Streichung dieser Variablen aus dem Forth-Vocabulary. Dadurch wird die Initialisierung eines Forth-Programms auf acht Byte gekürzt (SI-Register laden, AX löschen, Sprung nach »main«). Ab dieser Version befindet sich also bei Offset 0108hex der Beginn der Dictionary-Link-Kette, in Offset 0106hex steht die relative Entfernung von »main« (kann für Programme nützlich sein).
Neu eingeführt ist das Systemwort »"« (einzelnes Anführungszeichen). Bei Verwendung dieses Wortes wird Code generiert, um einen folgenden String im Dictionary zu überspringen, wobei die Anfangsadresse dieses Strings zur Laufzeit auf den Stack gelegt wird. Die Handhabung dieses Wortes ist analog zu »."«, allerdings wird der String nicht gedruckt, sondern seine Adresse lediglich auf den Stack gelegt.

Vocabulary Forth 1.3 (September 1987)
Die Systemvariablen »ds%«, »ss%«, »ax«, »bx«, »cx« und »dx« sind ersatzlos gestrichen. »cmove« und »fill« arbeiten jetzt nur noch innerhalb des Com-Segmentes. Neu aufgenommen sind die Worte »"« (Stringkonstante), »pick« und »roll« (beide wie in Forth 83), desgleichen »min« und »max« (liefern Minimum beziehungsweise Maximum der beiden obersten Stack-Werte). Das Wort »get« (Hilfswort zu »expect«) ist umbenannt in »(get)«, um anzudeuten, daß es nicht für die allgemeine Verwendung gedacht ist. »loop + « ist umbenannt in » + loop«.

SForth Version 2.4 (Oktober 1987)
Alle Diskettenoperationen, die der Compiler durchführt, sind auf untypisierte Dateien umgestellt und arbeiten nun mit BlockRead- beziehungsweise mit BlockWrite-Befehlen (Jörg Honsbrok). Dadurch wird ein drastischer Geschwindigkeitszuwachs bei »include« und bei der Com-File-Erzeugung erreicht.

Vocabulary Forth 1.4 (Dezember 1987)
Die grundlegenden Ein-/Ausgabe-Operationen werden über einen Vektor (Makro »(vector)« abgewickelt. Durch Umleiten dieses Vektors kann die Ein- und Ausgabe auf eigene Routinen gelegt werden.

Vocabulary Forth 1.5 (Januar 1988)
Das Forth-Vocabulary wurde nochmals überarbeitet und im Hinblick auf die Funktionsfähigkeit mit den Compiler-Versionen ab 3.0 getestet.

SForth Version 3.0 (Januar 1988)
Der Compiler steht in einer völlig neu überarbeiteten Version zur Verfügung (Erwin Schomburg). Wichtigstes Merkmal des neuen Compilers ist die Fähigkeit, auch speicherresidente Utilities damit erzeugen zu können. Eine ganze Reihe von Systemworten stehen für diesen Zweck im Compiler selbst und im Vocabulary »isys« zur Verfügung. Mit kleinen Überarbeitungen wurde daraus die derzeit aktuelle Version 3.1. Eine genaue Beschreibung dieser SForth-Version befindet sich im DOS Extra 2.

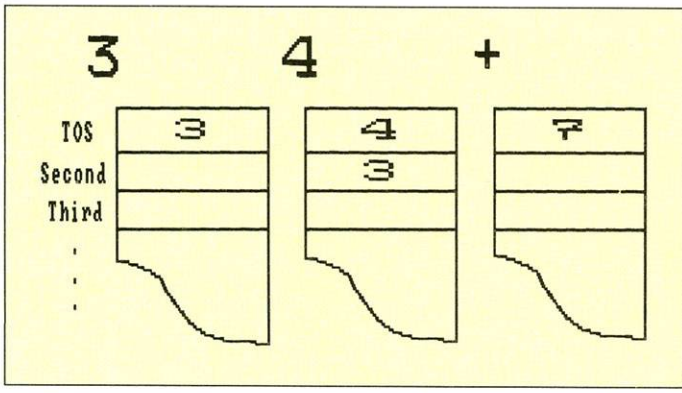


Bild 2. Der Datenstack während einer einfachen Rechnung

gen. Dieser Code hat für jede Konstante eine Länge von sieben Byte. Diese sieben Byte werden bei jeder einzelnen Nennung einer echten Konstanten benötigt.

Wenn allerdings eine Konstanten-Definition mit dem Schlüsselwort »constant« verwendet wird, dann entsteht der gleiche Code, allerdings mit einem zusätzlichen Return-Befehl. In diesem Fall wird zusätzlich analog zur Prozedur- oder Makrodefinition ein kompletter Wortrahmen aufgebaut, der den Namen der Konstanten und Angaben zum Typ enthält. Wenn die Konstante innerhalb eines anderen Wortes benötigt wird, zeigen sich die Vorteile dieses Verfahrens: Jeder Aufruf einer so definierten Konstanten wird in einen Call-Befehl übersetzt und benötigt daher nur noch drei Byte anstelle der sieben Byte für jede echte Konstante. Ein weiterer Vorteil: Wenn man wichtige vom Programm benötigte Zahlenwerte durch Konstanten repräsentiert, dann braucht man bei einer Änderung eines solchen Zahlenwertes nur eine einzige Programmänderung, nämlich in der Konstanten-Definition, durchzuführen. Eine solche Konstantendefinition sieht in SForth so aus:

```
constant TestConst 7
```

Die Wahl aussagekräftiger Konstantennamen gestaltet die Programme übersichtlicher. Groß- oder Kleinschreibung spielt hier – wie auch generell im gesamten Compiler – keine Rolle.

Variablen werden in gewisser Weise ähnlich behandelt wie Konstanten. Grundsätzlich arbeitet SForth auch in der Version 3.1 nur mit Integer-Variablen. Bei der Variablen-Definition erzeugt SForth Code, um die Adresse, unter der der Wert der Variablen abgelegt ist, auf den Stack zu legen. Diese Adresse dient dann als Ausgangspunkt für spezielle Forth-Befehle, die den Inhalt der Variablen manipulieren. Variablen werden in SForth generell mit einem Anfangswert versehen. Die Definition einer Variablen »TestVar« mit einem Initialisierungswert von 7 sieht beispielsweise so aus:

```
variable TestVar 7
```

Auf den ersten Blick ähnelt dies der Konstantendefinition. Es gibt aber einen ganz wesentlichen Unterschied im Gebrauch: Wenn der Name einer mit »constant« vereinbarten Konstanten in einer Definition vorkommt, wird zur Laufzeit der Wert dieser Konstanten auf den Stack gelegt. Wenn dagegen der Name einer Variablen vorkommt, wird die Adresse dieser Variablen (und nicht deren Inhalt) auf den Stack gelegt. Diesen Unterschied berücksichtigen Forth-Anfängern häufig nicht, was dann zu schwerwiegenden Programmfehlern führt.

Um nun den Inhalt einer Variablen zugänglich zu machen, wird das Wort »@« (sprich: (englisch) fetch) verwendet. »@« ersetzt eine auf dem Stack liegende Adresse durch deren (Integer-) Inhalt. Um den Inhalt der Variablen »alpha« aus unserem Beispiel anzuzeigen, kann man also folgende Definition verwenden:

```
: ZeigeAlpha alpha @ . ;
```

Mit dem zusätzlichen Hauptprogramm

```
: main ZeigeAlpha ;
```

können Sie das sofort ausprobieren. Der umgekehrte Fall liegt vor, wenn man einen Zahlenwert in einer Variablen ablegen will. Dazu dient das Wort »!« (sprich: (englisch) store). Dieses Wort speichert den Wert im Second (zweites Stackelement) an der Adresse im TOS (Top of Stack, oberstes Stapelelement). Um also beispielsweise der Variablen »alpha« den Wert 7 zuzuweisen, verwendet man die folgende Befehlsfolge:

```
7 alpha !
```

Obwohl Variable in vielen Fällen sehr nützlich sind, gilt es unter Forth-Programmierern als guter Stil, Variablen weitgehend zu vermeiden und sie nur für die langfristige Datenhaltung zu benutzen. Zwischenergebnisse sollten statt in Variablen immer auf dem Stack gehalten werden. SForth stellt daher eine Reihe von Befehlen zur komfortablen Manipulation des Datenstacks zur Verfügung. Ein Worttyp wurde bisher noch nicht besprochen, nämlich die System-Worte. Dabei handelt es sich um Worte, die der Compiler intern, also von sich aus, bereits kennt. Die System-Worte zerfallen in zwei Gruppen. Die eine Gruppe wird von den Worten gebildet, die Definitionen einleiten oder Compiler-Optionen darstellen. Diese Art von Worten darf innerhalb von Definitionen nicht verwendet werden. Die zweite Gruppe besteht aus Steuerworten, die die Programmstruktur kontrollieren. Die große SForth-Referenztabelle gibt Ihnen einen Überblick auch über diese Worte. Um die Referenztabelle richtig lesen zu können, hier zunächst einmal einige Grundlagen über die in Forth üblichen Stack-Diagramme. In Forth ist es üblich, die Funktionsweise von Worten ganz grob durch ein sogenanntes Stack-Diagramm zu verdeutlichen. Betrachten wir als Beispiel das Wort »swap«, dessen Definition im Forth-Vocabulary (Listing 2) beginnt mit

```
macro swap ( n1 n2 -> n2 n1 )
```

Der in Kommentarklammern stehende Ausdruck hinter »swap« stellt das Stackdiagramm dar. Links vom Pfeil wird immer der Stackzustand vor Aufruf des Wortes symbolisiert, rechts vom Pfeil befindet sich der Zustand nach Aufruf des Wortes. »n1« und »n2« stehen dabei für zwei beliebige Integer-Werte. Der TOS, also das oberste Stack-Element, steht immer rechts. Das Stackdiagramm zu »swap« muß man nun lesen als: Aus einer Zahlenfolge n1, n2 auf dem Stack macht »swap« die Zahlenfolge n2, n1. Mit anderen Worten: »swap« vertauscht die obersten beiden Stackelemente miteinander. Natürlich kann der Stack bei Aufruf von »swap« noch beliebig viele weitere Werte enthalten. Im Stackdiagramm werden grundsätzlich nur die Stack-Werte aufgeführt, die für die jeweilige Funktion relevant sind. Tabelle 1 zeigt in Stackdiagrammen häufig verwendete Symbole.

Schleifen und Verzweigungen: mit SForth strukturiert programmieren

SForth kennt keinen Goto-Befehl und auch keinen Ersatz dafür. Planloses Herumspringen im Programm und der daraus resultierende »Spaghetticode« wird nicht unterstützt. Dafür stehen eine Vielzahl von Befehlen zur strukturierten Ablaufsteuerung zur Verfügung. Diese Strukturen arbeiten allerdings ganz in Forth-Manier ebenfalls mit dem Datenstack. Betrachten wir zum Beispiel das

Definitions-Worte und Compiler-Optionen

(	(– >)	Beginn eines Kommentartextes
)	(– >)	Ende eines Kommentartextes
(DUMP)	(– >)	erzeugt einen Dump des aktuellen Dictionaries
(LIST.OFF)	(– >)	schaltet Text-Listing aus
(LIST.ON)	(– >)	schaltet Text-Listing während der Compilation ein
(SAVE.VOC)	(– >)	speichert Vocabulary in VOC-Datei
:	(– >)	leitet eine Prozedur-Definition ein
;	(– >)	beendet die laufende Definition
CONSTANT	(– >)	leitet eine Konstanten-Definition ein
INCLUDE	(– >)	fügt VOC-Datei in aktuellen Code ein
MAKRO	(– >)	leitet eine Makro-Definition ein
VARIABLE	(– >)	leitet eine Variablen-Definition ein
VOCABU-LARY	(– >)	markiert Beginn eines Vocabularies

System-Worte

(")	(– > adr)	Legt nachfolgenden String im Programmcode ab. Wird von »,"« aufgerufen und sollte nicht direkt verwendet werden
[	(– >)	Beginn von Inline-Code
]	(– >)	Ende von Inline-Code
BEGININT	(– >)	Start einer Interrupt-Service-Routine
CALLTO	(adr – >)	ruft eine andere Interrupt-Service-Routine auf
ENDINT	(– >)	Ende einer Interrupt-Service-Routine
EXITTO	(adr – >)	springt zu anderer Interrupt-Service-Routine
FIND	(– > adr)	liefert Dictionary-Adresse des direkt folgenden Wortes
INT	(n – >)	Software-Interrupt Nummer n aufrufen

Kontrollstrukturen

IF	(n – >)	Code zwischen »if« und »endif« wird ausgeführt, falls n = 0, sonst übersprungen
ELSE	(– >)	wird optional innerhalb einer If-Anweisung verwendet. Der Code zwischen »else« und »endif« wird ausgeführt, falls TOS < > 0
ENDIF	(– >)	Ende einer If-Anweisung
CASE	(– >)	Beginn einer mehrfachen Auswahl
OF	(n – >)	leitet Ausführungsteil in einer Case-Anweisung ein
ENDOF	(– >)	Abschluß einer Case-Fallunterscheidung
;;	(– >)	Synonym zu »endoff«
ENDCASE	(– >)	schließt den Ausführungsteil in einer Case-Anweisung ab
BEGIN	(– >)	Start einer »begin...until«-Schleife
AGAIN	(– >)	erzeugt eine »begin...again«-Schleife ohne Abbruchbedingung
REPEAT	(– >)	Ende einer »begin...while...repeat«-Schleife. Es wird zu »begin« zurückgesprungen
UNTIL	(n – >)	die Schleife wird ab »begin« ausgeführt, solange bei »until« n = 0
WHILE	(n – >)	wird innerhalb einer »begin...while...repeat«-Schleife verwendet. Der Code zwischen »while« und »repeat« wird nur ausgeführt, wenn n < > 0
DO	(n1 n2 – >)	Beginn einer Zählschleife. n2 enthält den Anfangswert, n1 den Endwert der Schleife
LOOP	(– >)	erhöht den Schleifenzähler um 1 und beendet die Schleife, wenn der Zähler größer als der Endwert ist
+LOOP	(n – >)	erhöht den Schleifenzähler um n, sonst wie »loop«
-LOOP	(n – >)	erniedrigt den Schleifenzähler um n, sonst wie »loop«
/LOOP	(u – >)	wie »+loop«, aber der Schleifenzähler wird als vorzeichenlose Zahl verwendet

System-Variablen

adr	(– > adr)	String-Adresse
base	(– > adr)	Zahlenbasis
dp	(– > adr)	freier Speicher
dpl	(– > adr)	Zeiger auf Dezimalpunkt
hld	(– > adr)	Zeiger auf Ziffer
in	(– > adr)	Zeichenzähler Eingabe
leading	(– > adr)	enthält führende Null oder Leerzeichen
ml	(– > adr)	maximale String-Länge
out	(– > adr)	Zeichenzähler für EMIT
r0	(– > adr)	Startadresse Return-Stack
s0	(– > adr)	Startadresse Daten-Stack
tib	(– > adr)	Terminal Input Buffer

System-Konstanten

0 0	(– > n)	Null
1 1	(– > n)	Eins
2 2	(– > n)	Zwei
bl	(– > n)	Leerzeichen
bs	(– > n)	Backspace
dictionary	(– > n)	Start Dictionary-Link
false	(– > n)	False-Flag
true	(– > n)	True-Flag

Stack-Manipulation

-dup	(n – > n ?)	n kopieren, wenn ungleich Null
>r	(n – >)	n auf Return-Stack speichern
ddup	(d – > d d)	doppeltgenaue Zahl kopieren
drop	(n – >)	n entfernen
dup	(n – > n n)	n kopieren
over	(n1 n2 – > n1 n2 n1)	n1 kopieren
r	(– > n)	TOS von Return-Stack kopieren
r>	(– > n)	TOS von Return-Stack holen
rot	(n1 n2 n3 – > n2 n3 n1)	zyklisches Vertauschen
swap	(n1 n2 – > n2 n1)	TOS und SECOND vertauschen

einfache Arithmetik

*	(n1 n2 – > n)	Multiplikation
*/	(n1 n2 n3 – > n)	berechnet n1*n2/n3
*/mod	(n1 n2 n3 – > r n)	berechnet Rest und n1*n2/n3
+	(n1 n2 – > n)	Addition
-	(n1 n2 – > n)	Subtraktion
/	(n1 n2 – > n)	Division
/mod	(n1 n2 – > r q)	Divisonsrest und Quotient
1+	(n – > n1)	TOS um 1 erhöhen
1-	(n – > n1)	TOS um 1 erniedrigen
2+	(n – > n1)	TOS um 2 erhöhen
2-	(n – > n1)	TOS um 2 erniedrigen
and	(n1 n2 – > n)	And-Verknüpfung
minus	(n – > -n)	Vorzeichenwechsel
mod	(n1 n2 – > r)	Divisionsrest
not	(n – > n1)	Not-Funktion
or	(n1 n2 – > n)	Or-Verknüpfung
xor	(n1 n2 – > n)	Xor-Verknüpfung

Double, Unsigned, Mixed Arithmetik

d+	(d1 d2 – > d)	doppeltgenaue Summe
m*	(n1 n2 – > d)	doppeltgenaues Produkt mit Vorzeichen
m/	(d n – > r q)	doppeltgenaue Division mit Vorzeichen
m/mod	(ud u – > r dq)	Rest und doppeltgenauer Quotient
u*	(u1 u2 – > ud)	doppeltgenaues Produkt ohne Vorzeichen
u/	(ud u – > Rest Quotient)	doppeltgenaue Division ohne Vorzeichen

Speicher-Befehle

!	(n adr – >)	Zahl speichern
+!	(n adr – >)	Zahl im Speicher addieren
++	(adr – >)	Speicherwort um 1 erhöhen
,	(– >)	Ein Speicherwort allokieren
–	(adr – >)	Speicherwort um 1 erniedrigen
@	(adr – > n)	Zahl holen
allot	(n – >)	Speicherplatz auf dem Heap allokieren
blanks	(adr n – >)	Speicherbereich mit Leerzeichen füllen
c!	(c adr – >)	Zeichen speichern
c,	(– >)	Ein Zeichen Speicherplatz allokieren
c@	(adr – > c)	Zeichen holen
cmove	(von bis n – >)	Speicherbereich verschieben
d!	(d adr – >)	doppeltgenauen Wert in Speicher schreiben
d@	(adr – > d)	doppeltgenauen Wert aus Speicher holen
erase	(adr n – >)	Speicherbereich mit Null füllen
fill	(adr n c – >)	Speicherbereich mit Zeichen c füllen
here	(– > adr)	Nächste freie Stelle auf dem Heap
pad	(– > adr)	Adresse des PAD-Bereiches
pick	(i – > ni)	Wert aus Datenstack holen
roll	(i – > ni)	Stack um i Elemente rotieren
rp!	(adr – >)	Returnstackpointer setzen
rp@	(– > adr)	Returnstackpointer holen
sp!	(adr – >)	Datenstackpointer setzen
sp@	(– > adr)	Datenstackpointer holen
toggle	(adr n – >)	Xor-Verknüpfung Speicherstelle

Unterstützung strukturierter Befehle

exit	(– >)	Wort abbrechen
i	(– > n)	Schleifenindex nach TOS kopieren
j	(– > n)	zweiter Schleifenindex
leave	(– >)	Schleife verlassen

Vergleichsoperatoren

<	(n1 n2 – > flag)	Test, ob n1 < n2
< >	(n1 n2 – > flag)	Test, ob n1 < > n2
=	(n1 n2 – > flag)	Test, ob n1 = n2
>	(n1 n2 – > flag)	Test, ob n1 > n2
0<	(n – > flag)	Test, ob n < 0
0< >	(n – > flag)	Test, ob n < > 0
0=	(n – > flag)	Test, ob n = 0

Funktionen

+-	(n1 n2 – > n)	berechnet sgn[n2]*n1
abs	(n – > abs)	Absolutwert
d+-	(d n – > n1)	berechnet sgn[n]*d
dabs	(d – > ud)	doppeltgenauer Absolutwert
dminus	(d – > -d)	Vorzeichenwechsel, doppeltgenau
s->d	(n – > d)	einfachgenaue Zahl in doppeltgenaue konvertieren
sgn	(n – > sgn)	Signum-Funktion

einfache Ein-/Ausgabe

"	(– > adr)	Adresse von String-Konstante holen
(app)	(– >)	Zeichen an Eingabestring anfügen
(bs)	(– >)	Backspace bei Eingabestring durchführen
(get)	(– > c)	Zeichen von Tastatur für Eingabestring holen
(vector)	(– >)	für vektorisierte Ein-/Ausgabe
.	(– >)	Text ausgeben
?key	(– > flag)	Taste gedrückt ?
count	(adr1 – > adr2 n)	String für Type vorbereiten
cr	(– >)	Zeilenvorschub ausführen
emit	(c – >)	TOS als ASCII-Zeichen ausgeben
expect	(adr count – >)	String einlesen
inkey	(– > c)	Tastatureingabe – > TOS; mit Echo
key	(– > c)	Tastatureingabe – > TOS
query	(– >)	Zeile in TIB einlesen
space	(– >)	Leerzeichen ausgeben
spaces	(n – >)	n Leerzeichen ausgeben
type	(adr count – >)	String ausgeben

formatierte Ausgabe

< #	(– >)	Start formatierte Ausgabe
#	(d1 – > d2)	eine Ziffer ablegen
#s	(d – > 0 0)	restliche Ziffern ablegen
# >	(d – > adr u)	Ende formatierte Ausgabe
.	(n – >)	einfach genaue Zahl ausgeben
.r	(n w – >)	einfach genaue Zahl in Feldbreite w ausgeben
d.	(d – >)	doppelt genaue Zahl ausgeben
d.r	(d w – >)	doppelt genaue Zahl in Feldbreite w ausgeben
hold	(c – >)	Zeichen an Adresse in hld speichern
sign	(n d – > d)	Vorzeichen in formatierte Ausgabe einfügen
u.	(u – >)	vorzeichenlose Zahl ausgeben
u.r	(u w – >)	vorzeichenlose Zahl in Feldbreite w ausgeben

verschiedene Funktionen

(number)	(d1 adr1 – > d2 adr2)	Zeichenfolge in Zahl wandeln
.s	(– >)	Stack ausdrucken
?	(adr – >)	Inhalt eines Speicherworts ausgeben
char	(adr1 – > adr2 c)	Zeichen holen, Leerzeichen überlesen
decimal	(– >)	Zahlenbasis 10 setzen
digit	(c – > ? flag)	Zeichen in Zahl konvertieren
hex	(– >)	Zahlenbasis 16 setzen
max	(n1 n2 – > max)	Maximum von n1, n2
min	(n1 n2 – > min)	Minimum von n1, n2
number	(adr – > d)	String in doppeltgenaue Zahl konvertieren

Referenztabelle SForth

Wort »if«. Durch dieses Wort wird der Wert des TOS getestet. Ist der TOS ungleich Null, dann wird der Programmteil zwischen »if« und dem dazugehörenden »endif« ausgeführt. War der TOS jedoch gleich Null, dann wird dieser Teil übersprungen und das Programm nach »endif« fortgesetzt. Auch eine Variante mit »else« ist vorgesehen:

```
if 1 else 2 endif
```

Die Befehlsfolge legt die Zahl 1 auf den Stack, wenn der TOS ungleich Null ist, sonst die Zahl 2. Generell entfernen alle Forth-Worte, die den Stack abfragen, den TOS vom Datenstack. Dadurch wird erreicht, daß der Stack sich im Laufe der Zeit nicht mit irgendwelchem Datenmüll füllt. Wenn der jeweilige TOS auch nach einer If-Abfrage noch gebraucht wird, dann muß er zuvor mit dem Wort »dup« dupliziert werden. Dieses Wort ist im Forth-Vocabulary enthalten.

Neben der Verzweigung sind Programmschleifen ein wichtiger Bestandteil jeder strukturierten Sprache. SForth bietet hier eine breite Auswahl an. Die einfache »begin...until«-Schleife wird solange durchlaufen, bis der TOS beim Aufruf von »until« ungleich Null ist. Auch »until« entfernt den TOS vom Datenstack. Eine Variante dieser Schleifenart ist die Endlosschleife »begin...again«. Hier wird keine Bedingung getestet, sondern die Schleife läuft endlos weiter – wenn sie nicht durch besondere Maßnahmen (beispielsweise einen Exit-Befehl) verlassen wird.

Eine weitere Variante ist etwas aufwendiger, bietet in der Programmierpraxis jedoch vielfältige Möglichkeiten. Die Rede ist von der Begin...while...repeat-Schleife. Bei dieser Schleife wird die Bedingung innerhalb des Schleifenkörpers bei »while« getestet. Der Programmteil zwischen »begin« und »while« wird auf jeden Fall ausgeführt, der zwischen »while« und »repeat« jedoch nur dann, wenn der TOS bei »while« einen Wert ungleich Null hat. Anderfalls wird sofort die Schleife verlassen und die Ausführung des Programms hinter »repeat« fortgesetzt. Solange die Abbruchbedingung bei »while« nicht erfüllt ist, verzweigt »repeat« immer wieder an den Schleifenanfang zu »begin«.

Als letzte Schleifenkonstruktion ist die »do...loop«-Schleife vorhanden. Es handelt sich dabei um eine Zählschleife, die von einem Anfangswert zu einem Endwert zählt. Diese beiden Werte müssen beim Aufruf von »do« auf dem Stack liegen, und zwar der Anfangswert als TOS und der Endwert als Second. »Do« entfernt beide Werte vom Datenstack und speichert sie intern. Diese Speicherung geschieht bei Standard-Forth auf dem Return-Stack. Bei SForth dagegen werden die Parameter der jeweils innersten »do...loop«-Konstruktion aus Effizienzgründen direkt in Prozessorregistern gehalten. Das Wort »loop« erhöht den internen Schleifenzähler um Eins und vergleicht mit dem Endwert der Schleife. Ist der Endwert überschritten, dann wird die Schleife verlassen. Eine Abwandlung ist mit dem Wort » + loop« möglich. Hier wird der Schleifenzähler um den Wert des TOS erhöht, wodurch Schleifen mit variabler Schrittweite möglich werden. » + loop« arbeitet ansonsten analog zu »loop«, die Schleife wird also bei Überschreiten des Endwerts verlassen. Der Schleifenzähler der Do-Schleife ist dem Programm über das Wort »i« aus dem Forth-Vocabulary zugänglich.

Ein wichtiger Bestandteil des Compilers wurde noch nicht erwähnt: Durch das Wort »[« wird Inline-Code in eine Makro- oder Prozedur-Definition eingefügt. Die gesamte in eckige Klammern eingeschlossene Zahlenfolge (dezimal oder hexadezimal) wird direkt in den Programmcode compiliert. Die Inline-Anweisung bildet die Grundlage für die Definition vieler maschinennaher Worte des Forth-Vocabulary.

Der Kern des SForth-Compilers ist in Turbo Pascal geschrieben

SForth wird, wie bereits zu Anfang bemerkt, als reiner Compiler betrieben. Der Compiler selbst ist in Turbo Pascal geschrieben (Listing 1). Er übersetzt lediglich die System-Worte von SForth (siehe Referenztabelle SForth). Dies reicht aber bereits aus, um das vollständige SForth-System gewissermaßen »aus sich selbst heraus« hochzuziehen. Denn aufbauend auf den System-Worten werden alle weiteren Worte des SForth-Systems in SForth selbst geschrieben, wobei die ersten dieser Worte vielfach unter Verwendung von Inline-Codierung programmiert werden.

Um den Compiler einzusetzen, ist Turbo Pascal Version 3.0 erforderlich. Das Programm ist übersichtlich und dokumentiert sich weitgehend selbst. Um in SForth geschriebene Programme zu übersetzen, stellen Sie als erstes eine Com-Version des als Listing abgedruckten Compilers her. Wählen Sie dazu aus dem Optionen-Menü von Turbo Pascal die C-Option aus und compilieren Sie das Programm als »sf.com«. Schon ist Ihr SForth-Compiler fertig. Alternativ können Sie natürlich auch den übersetzten und lauffähigen Compiler aus dem Databox-Service zu diesem DOS Extra verwenden. Mit dem fertigen Com-Programm können Sie nun bereits etwas experimentieren. Dazu erzeugen Sie zum Beispiel (am einfachsten mit dem Turbo Pascal-Editor) eine Textdatei namens »test.fth«, die einen einfachen Forth-Text enthält, beispielsweise:

```
( ein erster Test )
: main 9 8 7 ;
```

Dieses kleine Programm legt die Zahlen 9, 8 und 7 auf den Datenstack (die Zahlen 0, 1 und 2 sollten Sie nicht verwenden, die sind nämlich als Konstanten im Forth-Vocabulary definiert). Davon sehen Sie aber nichts, weil noch keine Ein-/Ausgabebefehle programmiert wurden, die man verwenden könnte. Betrachten Sie das Ganze daher nur als kleinen Test der Funktionsfähigkeit des Compilers. Immerhin können Sie die Compiler-Optionen ausprobieren. Die Übersetzung des kleinen Testprogramms von vorhin erfolgt durch den Aufruf »sf test«. Bei diesem Aufruf wird während der Compilierung lediglich die laufende Zeilennummer angezeigt.

Sie steuern die Ausgabe durch Angabe von Optionen. Optionen werden, durch jeweils ein Leerzeichen voneinander getrennt, hinter dem Programmnamen angegeben. Eine Option besteht aus dem Zeichen » – « oder »/«, gefolgt von einem Kennbuchstaben. Zur Verfügung stehen folgende Routinen:

– L: Es wird während der Compilation ein komplettes Listing des Quelltextes mit Zeilennummern ausgegeben. Wenn der Quelltext die System-Worte »(list.on)« oder »(list.off)« enthält, so haben diese allerdings Vorrang vor der L-Option.

– P: Es wird der jeweilige Stand des Programmzählers zu jeder neuen Zeile in hexadezimaler Form ausgegeben.

– D: Bei einem Fehlerabbruch wird automatisch ein Dump des Dictionary ausgegeben. Eine Zeile des Dumps enthält den Namen des Wortes, den Typ als Kennziffer und den Code in hexadezimaler Form. Der Dictionary-Dump kann als Hilfsmittel bei der Fehlersuche eingesetzt werden.

– N: Diese Option bewirkt, daß zum Abschluß der Compilation kein Com-File mit dem übersetzten Forth-Programm erzeugt wird. Der N-Schalter ist für Testläufe gedacht.

– Mn: Mit dieser Option wird der Speicherbedarf für den Laufzeit-Heap festgelegt.

SForth arbeitet normalerweise in einem einzigen 64 KByte-Segment, wobei Programm und Daten im unteren Segment-Bereich abgelegt werden, und die beiden Stack-Bereiche an hohen Speicheradressen stehen. Der Bereich dazwischen ist für einen Heap-Bereich vorgesehen, der sich während der Programmausführung frei verwenden läßt. Die M-Option ermöglicht es also, die Größe dieses Bereichs zu ändern. Für Ihre normalen Programme brauchen Sie diese Option nie einzusetzen. Wenn Sie allerdings speicherresidente Utilities in SForth programmieren, dann können Sie den vom Programm benötigten Speicherplatz damit drastisch reduzieren.
Zusätzlich haben Sie die Möglichkeit, alle Ausgaben des Programms wie unter MS-DOS üblich in beliebige Dateien oder Geräte umzuleiten. Beispielsweise wird durch den Aufruf

```
sf test -l -n >prn
```

ein Listing des Forth-Textes auf dem Drucker erzeugt.
Mit diesem Compiler haben Sie bereits die Grundlage für das SForth-System in der Hand. Echte SForth-Programme können Sie aber erst mit dem im folgenden beschriebenen Forth-Vocabulary herstellen.

Das gesamte Standard-Vocabulary des SForth-Compilers ist in SForth geschrieben

Die hier vorgestellte Standardbibliothek zu SForth (Listing 2) ist ein integraler Bestandteil des SForth-Compilers und in Forth-üblicher Form als sogenanntes »Vocabulary« organisiert. Um das Forth-Vocabulary in eigene SForth-Programme einzubinden, setzen Sie die Zeile

```
include forth
```

an den Anfang Ihres Listings. Danach steht es Ihnen frei, alle im Forth-Vokabulary enthaltenen Worte zu verwenden. Im Prinzip ist es möglich, das Forth-Vocabulary zu erweitern oder zu ändern, im Interesse einer weitgehenden Portabilität von SForth-Programmen sollten Sie aber darauf verzichten und statt dessen eigene, zusätzliche Vocabularies für spezielle Anwendungen schreiben. Dies ist sehr einfach möglich: Durch die Zeile

```
vocabulary MeinVoc
```

definieren Sie beispielsweise den Beginn eines Vokabularies mit dem Namen »MeinVoc«. Alle im folgenden definierten Worte gehören dann zu diesem Vocabulary. Mit der Anweisung

```
(save.voc)
```

veranlassen Sie den Compiler, das jeweils zuletzt benannte Vocabulary in bereits übersetzter Form, also als Binärdatei zu speichern. Durch eine Include-Anweisung wird diese Binärdatei wieder eingebunden, wobei allerdings geprüft wird, ob der enthaltene Code auch an die richtige Adresse geladen wird. Wenn Sie also zwei Vocabularies einbinden und anschließend am ersten noch Änderungen vornehmen, dann müssen Sie beide Vocabularies neu übersetzen, da das zweite durch die Änderungen am ersten einen anderen Speicherbereich belegt.
Bei der Durchsicht des SForth-Vocabularies werden Sie feststellen, daß SForth sich stark am Forth-Standard orientiert. Es fehlen lediglich einige Bestandteile der Sprache, die für einen Interpreter gedacht sind und mit einem Compiler nicht zu realisieren sind. Die Referenztabelle zu SForth zeigt alle Forth-Worte, die vom Compiler zusammen mit dem Forth-Vocabulary unterstützt werden.
Zur Programmierung ist noch ein Wort zu sagen: SForth erzeugt bei der Übersetzung echte Com-Dateien, wobei das auszuführende Wort (das Hauptprogramm) den Namen »main« haben muß. Damit die Eingabe des Forth-Vocabularies und das Erstellen eigener Programme möglichst problemlos abläuft, sollten Sie folgende Punkte beherzigen:

- Die Zeilennummern in den abgedruckten Listings dienen nur zur Referenz. SForth arbeitet nicht mit Zeilennummern. Geben Sie diese Nummern also auf keinen Fall mit ein.
- Kommentarklammern und die eckigen Klammern für Inline-Code sind jeweils eigene Forth-Worte. Achten Sie also darauf, daß zwischen Klammer und folgendem Wort immer mindestens ein Leerzeichen steht.
- Einige Worte enthalten Klammern als Teil des Namens, zum Beispiel »(save.voc)«. Solche Namen sind laut Forth-Konvention für Compiler-Optionen oder andere, nicht für die direkte Programmierung vorgesehene Worte gedacht. In diesem Falle stellen die Klammern also keinen Kommentar dar und dürfen auf keinen Fall durch ein Leerzeichen vom übrigen Wort getrennt werden.

Mehr über SForth und vor allem eine ganze Reihe von interessanten Beispielprogrammen und zusätzliche Vocabularies finden Sie in den folgenden Beiträgen zum SForth-Projekt.

(ev)

```
[XF5]    0: program SForth;
[000]    1:
[K49]    2: { S-Forth-Compiler Vers. 3.1 }
[000]    3:
[BRD]    4: { (c) 1987, 1988 Volker Everts & Erwin Scho
            mburg }
[000]    5:
[V30]    6: {
[R8C]    7: ** SForth ist ein Software-Projekt von
[EK5]    8: ** DOS International
[Y30]    9: **
[ZNB]   10: ** Dokumentation siehe DOS Extra 2, 1988
[LWC]   11: ** Die ursprüngliche Version des Compilers
[SKB]   12: ** wurde in den DOS-Ausgaben 9/87 bis 11/87
[8Y5]   13: ** veroffentlicht.
[Y30]   14: **
[1VC]   15: ** Seither wurde der SForth-Compiler unter
[9PB]   16: ** Mitwirkung der Leser von DOS Inter-
[01D]   17: ** national stark modifiziert, verbessert
[NPD]   18: ** und ausgebaut. Diese neueste Version 3.1
[COB]   19: ** des Compilers ist in erster Linie
[ZAD]   20: ** Erwin Schomburg zu verdanken, der durch
```

```
[H8D]   21: ** viele Vorschläge und Anregungen ent-
[12D]   22: ** scheidenden Einfluß auf die Weiterent-
[7CB]   23: ** wicklung des SForth-Systems hatte.
[X30]   24: }
[000]   25:
[000]   26:
[VM1]   27: const
[M4A]   28:   VERSION   = '3.1'; { Versionsnummer  }
[269]   29:   MAXSTACK  = 100;   { Stack-Ebenen    }
[519]   30:   EMPTY     = '';    { leerer String   }
[HM8]   31:   SPACE     = #32;   { Leerzeichen     }
[E88]   32:   BEL       = #7;    { akust. Signal   }
[4M8]   33:   CR        = #13;   { Wagenrücklauf   }
[Z09]   34:   UNDEF     = 0;     { undefiniert     }
[DZ9]   35:   KONSTANTE = 1;     { Forth-Konstante }
[DD9]   36:   VARIABLE  = 2;     { Forth-Variable  }
[6K8]   37:   MAKRO     = 3;     { Forth-Makro     }
[AU9]   38:   PROZEDUR  = 4;     { Forth-Prozedur  }
[B89]   39:   SYSTEM    = 5;     { Systembefehl    }
[000]   40:
[000]   41:
[000]   42:
```

```
[A21]   43: type
[KVC]   44:   Memory = array[256..$7FFF] of byte; { 32
          KByte }
[RT6]   45:   WortTyp = string[16];
[467]   46:   ZeilenTyp = string[127];
[HP5]   47:   HexStr = string[4];
[RX5]   48:   StackEintrag = record
[B54]   49:                    Wert: integer;
[MM2]   50:                    Typ: byte;
[TUO]   51:                  end;
[LND]   52:   StackTyp = array[0..MAXSTACK] of StackEin
          trag;
[000]   53:
[000]   54:
[000]   55:
[EMO]   56: var
[TSB]   57:   infile: text;          { Forth-Quelltext
          }
[S8C]   58:   outfile: file;         { compilierter Code
          }
[BNC]   59:   Zeile, LZ: ZeilenTyp; { Forth-Textzeile
          }
[000]   60:
[L87]   61:   Wort,              { Forth-Wort }
[LN6]   62:   Name,              { Dateiname  }
[SN6]   63:   Ext,               { Extension  }
[7C8]   64:   TextFile,          { Quelldatei }
[XC8]   65:   VocName,           { Vocabulary }
[KF2]   66:   Merker,
[WU8]   67:   Merker2: WortTyp; { Merker     }
[000]   68:
[AXE]   69:   sinit,             { Merker für Stack-Init
          ialisierung }
[F28]   70:   addr,              { Hilfsadresse }
[G9B]   71:   maxallot,          { max. Speicher frei }
[UL8]   72:   pc,                { Programmzähler     }
[368]   73:   sp,                { Stackpointer       }
[DH9]   74:   Nummer,            { Zeilennummer       }
[N3A]   75:   Anfang,            { Anfang Dictionary  }
[TV8]   76:   Ende,              { Ende Dictionary    }
[M1B]   77:   Zeiger,            { auf aktuelles Wort }
[X5A]   78:   VocAnfang,         { Start Vocabulary   }
[9G7]   79:   ofcnt,             { OF-Zähler          }
[V18]   80:   i,                 { Zählvariable       }
[RVB]   81:   mjmp: integer;     { Sprungadresse Main }
[000]   82:
[TLA]   83:   M: Memory;        { Speicherbereich    }
[MMA]   84:   S: StackTyp;      { Kontroll-Stack     }
[000]   85:
[AH9]   86:   main,              { Hauptprogramm      }
[479]   87:   dup,               { für Optimierung    }
[B49]   88:   sys,               { System-Befehl      }
[BV2]   89:   caseflag,
[WGA]   90:   gefunden,          { für Wortsuche      }
[H8A]   91:   ListFlag,          { Listing ein/aus    }
[BEA]   92:   DumpFlag,          { Dump ein/aus       }
[LOB]   93:   NoCodeFlag,        { speichern ein/aus  }
[S9C]   94:   pcFlag: boolean; { pc zeigen ein/aus  }
[000]   95:
[000]   96:
[000]   97:
[THA]   98: function hex(n, l: integer): HexStr;
[000]   99:
[LVA]  100: { n in l-stellige Hexzahl wandeln }
[000]  101:
[EMO]  102: var
[DV3]  103:   i, z: integer;
[XK2]  104:   s: HexStr;
[000]  105:
[BH1]  106: begin
[532]  107:   s := EMPTY;
[GA4]  108:   for i := 1 to l do
[BH1]  109:     begin
[B58]  110:       z := n and 15; { Ziffer bilden }
[UH5]  111:       if z > 9 then z := z + 7;
[GB4]  112:       s := chr(z + 48) + s;
[T99]  113:       n := n shr 4; { Division durch 16 }
[TUO]  114:     end;
[5P1]  115:   hex := s;
[TUO]  116: end;
[000]  117:
[000]  118:
[000]  119:
[MD5]  120: procedure dump;
[000]  121:
[CX9]  122: { Dictionary als Dump ausgeben }
[000]  123:
[EMO]  124: var
[MV3]  125:   p, z: integer;
```

```
[CR3]  126:   cnt: integer;
[G53]  127:   s: WortTyp;
[000]  128:
[BH1]  129: begin
[SP2]  130:   p := Anfang;
[9Z2]  131:   writeln;
[8V7]  132:   writeln('***** Start Dump *****');
[X52]  133:   repeat
[RV1]  134:     cnt:=1;
[9Z2]  135:     writeln;
[6W3]  136:     move(M[p], z, 2);
[PE4]  137:     if not (z=0) then
[BH1]  138:     begin
[AP5]  139:     write(hex(z, 4), SPACE);
[PH4]  140:     move(M[p+2], s, 17);
[FN2]  141:     write(s);
[W66]  142:     for i:=length(s) to 13 do
[N73]  143:     write(SPACE);
[8F5]  144:     p := p + 3 + length(s);
[Z93]  145:     case m[p] of
[3X5]  146:     1:write('KONSTANTE ');
[AN5]  147:     2:write('VARIABLE  ');
[BE5]  148:     3:write('MACRO     ');
[FU5]  149:     4:write('PROZEDUR  ');
[AV5]  150:     5:write('SYSTEM    ');
[EK5]  151:     else write('HUH?       ');
[TUO]  152:     end;
[6H4]  153:     write(M[p], SPACE);
[5X1]  154:     p := p + 1;
[L74]  155:     while p < z do
[BH1]  156:       begin
[G4A]  157:         if (cnt>1) and ((cnt mod 16) = 1) t
         hen
[BH1]  158:         begin
[9Z2]  159:         writeln;
[RL3]  160:         for i:=1 to 31 do
[N73]  161:         write(SPACE);
[TUO]  162:         end;
[BM6]  163:         write(hex(M[p],2), SPACE);
[4S4]  164:         cnt:=succ(cnt);
[5X1]  165:         p := p + 1;
[TUO]  166:       end;
[TUO]  167:   end;
[9T2]  168:   until z=0;
[9Z2]  169:   writeln;
[5DD]  170:   write('*** Ausgabe des fehlerhaften Reste
         s ***');
[RV1]  171:   cnt:=1;
[KA5]  172:   for i := p to pc-1 do
[BH1]  173:   begin
[EW8]  174:   if (cnt mod 16)=1 then writeln;
[8L6]  175:   write(hex(M[i],2), SPACE);
[4S4]  176:   cnt:=succ(cnt);
[TUO]  177:   end;
[9Z2]  178:   writeln;
[TUO]  179: end;
[000]  180:
[000]  181:
[000]  182:
[AK8]  183: procedure error(nr: byte);
[000]  184:
[YA6]  185: { Fehlerbehandlung }
[000]  186:
[976]  187: var msg: string[80];
[000]  188:
[BH1]  189: begin
[ZW2]  190:   case nr of
[5E7]  191:       1: msg := 'Syntax-Fehler';
[7H7]  192:       2: msg := 'Name erwartet';
[BB7]  193:       3: msg := 'doppelter Name';
[8H9]  194:       4: msg := 'undefinierter Name';
[62C]  195:       5: msg := 'Proc-Aufrufe nicht erlaubt'
         ;
[02D]  196:       6: msg := 'System-Befehle nicht erlaub
         t';
[6H9]  197:       7: msg := 'Zahlenwert erwartet';
[FN7]  198:       8: msg := 'Stack-Überlauf';
[Y97]  199:       9: msg := 'Stack ist leer';
[9H9]  200:      10: msg := 'ENDIF oder ELSE ohne IF';
[9MA]  201:      11: msg := 'SHORT-Option unmöglich';
[NEB]  202:      12: msg := 'unerwartetes Zeilenende';
[1N7]  203:      13: msg := 'fehlendes BEGIN';
[3T7]  204:      14: msg := 'fehlendes WHILE';
[WD6]  205:      15: msg := 'LOOP ohne DO';
[LBC]  206:      16: msg := 'Zeichenkonstante erwartet';
[BCP]  207:      17: msg := 'SF <source> [-L(isting) -P(
         c count) -D(ump) -N(o COM-File -M(axallot)]'
           ;
[PMF]  208:      18: msg := 'illegaler Parameter in Komm
         andozeile';
```

```
[BNA] 209:      19: msg := 'Dateifehler Quelltext';
[626] 210:      20: msg := 'IF ohne ENDIF';
[EDA] 211:      21: msg := 'BEGIN ohne UNTIL oder AGAIN
      ';
[L18] 212:      22: msg := 'WHILE ohne REPEAT';
[R16] 213:      23: msg := 'DO ohne LOOP';
[76C] 214:      24: msg := 'kein Vocabulary definiert';
[C7A] 215:      25: msg := 'Dateifehler bei INCLUDE';
[PX9] 216:      26: msg := 'falsche INCLUDE-Adresse';
[7S9] 217:      27: msg := ': MAIN Definition fehlt';
[20D] 218:      28: msg := 'CASE-Schachtelung nicht erl
      aubt';
[GEA] 219:      else msg := 'unbekannter Fehler';
[TU0] 220:    end;
[9Z2] 221:    writeln;
[M1A] 222:    if not ListFlag then writeln(LZ);
[NY5] 223:    writeln(BEL, '+++++ ', msg);
[Z3C] 224:    if DumpFlag and not (Nr in [17, 18, 19])
      then
[BH1] 225:      begin
[WB1] 226:        dump;
[9Z2] 227:        writeln;
[GK3] 228:        writeln(LZ);
[W14] 229:        writeln(msg);
[TU0] 230:      end;
[5B1] 231:    halt;
[TU0] 232: end;
[000] 233:
[000] 234:
[000] 235:
[000] 236:
[BD5] 237: procedure init;
[000] 238:
[3Z8] 239: { Compiler initialisieren }
[000] 240:
[EM0] 241: var
[BA2] 242:   p, i: byte;
[TP5] 243:   option: string[2];
[2D2] 244:   ch: char;
[000] 245:
[BH1] 246: begin
[B82] 247:   LZ := EMPTY;
[2Z8] 248:   if ParamCount = 0 then error(17);
[000] 249:
[W26] 250:   { Dateinamen holen }
[000] 251:
[1T6] 252:   TextFile := ParamStr(1);
[EW5] 253:   p := pos('.', TextFile);
[X13] 254:   if p = 0 then
[BH1] 255:     begin
[BR4] 256:       Name := TextFile;
[WV2] 257:       Ext := '.FTH';
[P57] 258:       TextFile := TextFile + Ext;
[DL0] 259:     end
[A11] 260:   else
[BH1] 261:     begin
[EU7] 262:       Name := copy(TextFile, 1, p-1);
[C57] 263:       Ext := copy(TextFile, p, 80);
[TU0] 264:     end;
[000] 265:
[SE9] 266:   { Options-Voreinstellungen }
[000] 267:
[K55] 268:   ListFlag := false;
[055] 269:   DumpFlag := false;
[C45] 270:   NoCodeFlag := false;
[BX4] 271:   pcFlag := false;
[SF3] 272:   maxallot:=0;
[000] 273:
[VK7] 274:   { Optionen auswerten }
[000] 275:
[NR6] 276:   if ParamCount > 1 then
[L77] 277:     for i := 2 to ParamCount do
[BH1] 278:       begin
[XF6] 279:         option := ParamStr(i);
[LU8] 280:         if option[1] in ['/', '-'] then
[BH1] 281:           begin
[7B6] 282:             ch := UpCase(option[2]);
[RS2] 283:             case ch of
[HS5] 284:               'L': ListFlag := true;
[1T5] 285:               'D': DumpFlag := true;
[D77] 286:               'N': NoCodeFlag := true;
[3F4] 287:               'P': pcFlag := true;
[K82] 288:               'M': begin
[N39] 289:                      Merker:=copy(paramstr(
      i), 3, 80);
[8V7] 290:                      val(Merker, maxallot,
      addr);
[RX6] 291:                      if (addr <> 0) then er
      ror(7);
[TU0] 292:                    end;
[044] 293:             else error(18);
[293] 294:           end; { case }
[DG2] 295:         end { if }
[1UA] 296:       else if option[1] <> '>' then error
      (18);
[2V2] 297:     end; { for }
[000] 298:
[DN8] 299:   { Quelltextdatei öffnen }
[000] 300:
[NA7] 301:   assign(infile, TextFile);
[5W6] 302:   {$I-} reset(infile); {$I+}
[948] 303:   if IOresult <> 0 then error(19);
[000] 304:
[X1A] 305:   { verschiedene Einstellungen }
[000] 306:
[644] 307:   Zeile := EMPTY;
[NG4] 308:   Merker := EMPTY;
[ZB3] 309:   Wort := SPACE;
[WU3] 310:   main := false;
[VY2] 311:   dup := false;
[WE1] 312:   sp := 0;
[HX2] 313:   Nummer := 0;
[TL1] 314:   pc := 256;
[VP4] 315:   VocName := EMPTY;
[2D2] 316:   ofcnt:=0;
[UH5] 317:   caseflag:=false;
[S14] 318:   VocAnfang := 0;
[TU0] 319: end;
[000] 320:
[000] 321:
[000] 322:
[LU8] 323: function HoleWort: WortTyp;
[000] 324:
[5S9] 325: { Ein Wort aus Quelltext holen }
[000] 326:
[093] 327: label leave;
[000] 328:
[EM0] 329: var
[4N2] 330:   p: byte;
[M53] 331:   w: WortTyp;
[000] 332:
[BH1] 333: begin
[X52] 334:   repeat
[NO6] 335:     if Zeile = EMPTY then
[0U4] 336:       if eof(infile) then
[BH1] 337:         begin
[UU4] 338:           HoleWort := EMPTY;
[T13] 339:           goto leave;
[DL0] 340:         end
[A11] 341:       else
[BH1] 342:         begin
[VZ5] 343:           readln(infile, Zeile);
[BG2] 344:           LZ := Zeile;
[BGB] 345:           if pcFlag then write(hex(pc,4), S
      PACE, SPACE);
[KX6] 346:           Nummer := succ(Nummer);
[BKE] 347:           if ListFlag then writeln(Nummer:5
      , '   ', Zeile)
[K27] 348:           else write(Nummer:5, CR);
[TU0] 349:         end;
[WLE] 350:     while copy(Zeile, 1, 1) = SPACE do dele
      te(Zeile, 1, 1);
[7B5] 351:   until zeile <> EMPTY;
[TO6] 352:   p := pos(SPACE, Zeile);
[X13] 353:   if p = 0 then
[BH1] 354:     begin
[ZD2] 355:       w := Zeile;
[644] 356:       Zeile := EMPTY;
[DL0] 357:     end
[A11] 358:   else
[BH1] 359:     begin
[DB6] 360:       w := copy(Zeile, 1, p-1);
[LS7] 361:       Zeile := copy(Zeile, p+1, 255);
[TU0] 362:     end;
[LH3] 363:   Merker2 := w;
[YAD] 364:   for p := 1 to length(w) do w[p] := UpCase
      (w[p]);
[435] 365:   dup := Merker = 'DUP';
[C33] 366:   Merker := w;
[974] 367:   HoleWort := w;
[6V1] 368:   Leave:
[TU0] 369: end;
[000] 370:
[000] 371:
[000] 372:
[N88] 373: function HoleZeichen: char;
[000] 374:
[YFE] 375: { Ein einzelnes Zeichen aus Quelltext holen
      }
```

```
[000]  376:
[BH1]  377: begin
[YG8]  378:   if Zeile = EMPTY then error(12);
[GR6]  379:   HoleZeichen := Zeile[1];
[UT6]  380:   Zeile := copy(Zeile, 2, 255);
[TUO]  381: end;
[000]  382:
[000]  383:
[000]  384:
[8H6]  385: procedure ob(b: Byte);
[000]  386:
[AX9]  387: { Ein Byte in Dictionary ablegen }
[000]  388:
[BH1]  389: begin
[VC2]  390:   M[pc] := b;
[OK2]  391:   pc := pc + 1;
[TUO]  392: end;
[000]  393:
[000]  394:
[000]  395:
[P48]  396: procedure ow(w: integer);
[000]  397:
[T2A]  398: { Ein Wort in Dictionary ablegen }
[000]  399:
[BH1]  400: begin
[Y94]  401:   move(w, M[pc], 2);
[4K2]  402:   pc := pc + 2;
[TUO]  403: end;
[000]  404:
[000]  405:
[000]  406:
[739]  407: procedure ot(adr, w: integer);
[000]  408:
[T2E]  409: { Ein Wort an spezifizierter Adresse ablege
       n }
[000]  410:
[BH1]  411: begin
[EW4]  412:   move(w, M[adr], 2);
[TUO]  413: end;
[000]  414:
[000]  415:
[000]  416:
[Z48]  417: procedure os(s: WortTyp);
[000]  418:
[5Z9]  419: { String im Dictionary ablegen }
[000]  420:
[BH1]  421: begin
[CTA]  422:   for i := 0 to length(s) do ob(ord(s[i]));
[TUO]  423: end;
[000]  424:
[000]  425:
[000]  426:
[938]  427: procedure TueKommentar;
[000]  428:
[YY7]  429: { Kommentar überlesen }
[000]  430:
[SS2]  431: var z: char;
[000]  432:
[BH1]  433: begin
[X52]  434:   repeat
[YR4]  435:     z := HoleZeichen;
[KM7]  436:   until (z = ')') or (Zeile = EMPTY);
[TUO]  437: end;
[000]  438:
[000]  439:
[000]  440:
[NM5]  441: procedure TueCode;
[000]  442:
[B28]  443: { Inline-Code auswerten }
[000]  444:
[E42]  445: label 99;
[000]  446:
[EMO]  447: var
[M53]  448:   w: WortTyp;
[G36]  449:   disp, Fehler: integer;
[000]  450:
[BH1]  451: begin
[X52]  452:   repeat
[OW3]  453:     w := HoleWort;
[D57]  454:     if w = EMPTY then error(7);
[446]  455:     if w = ']' then goto 99;
[405]  456:     val(w, disp, Fehler);
[T77]  457:     if Fehler <> 0 then error(7);
[LOC]  458:     if disp > 255 then ow(disp) else ob(dis
       p);
[TAO]  459:     99:
[TR3]  460:   until w = ']';
[TUO]  461: end;
```

```
[000]  462:
[000]  463:
[000]  464:
[ZGC]  465: procedure push(p: integer; Flag: Byte);
[000]  466:
[C8B]  467: { Adresse und Flag auf Stack ablegen }
[000]  468:
[BH1]  469: begin
[B84]  470:   S[sp].Wert := p;
[UD4]  471:   S[sp].Typ := Flag;
[KL2]  472:   sp := sp + 1;
[LV8]  473:   if sp > MaxStack then error(8);
[TUO]  474: end;
[000]  475:
[000]  476:
[000]  477:
[TTA]  478: function pop(Flag: Byte): integer;
[000]  479:
[ZBC]  480: { Adresse vom Stack holen, Flag prüfen }
[000]  481:
[BH1]  482: begin
[PL2]  483:   sp := sp - 1;
[P36]  484:   if sp < 0 then error(9);
[9HB]  485:   if S[sp].Typ = Flag then pop := S[sp].Wer
       t
[A11]  486:   else
[P63]  487:     case Flag of
[DB3]  488:       1: error(10);
[RB3]  489:       2: error(13);
[VB3]  490:       3: error(14);
[ZB3]  491:       4: error(15);
[OS3]  492:       else error(0);
[TUO]  493:     end;
[TUO]  494: end;
[000]  495:
[000]  496:
[000]  497:
[7RD]  498: function near(Quelle, Ziel: integer): integ
       er;
[000]  499:
[92A]  500: { Near-Sprungdistanz berechnen }
[000]  501:
[5L5]  502: var disp: integer;
[000]  503:
[BH1]  504: begin
[KT5]  505:   near := Ziel - Quelle - 2;
[TUO]  506: end;
[000]  507:
[000]  508:
[000]  509:
[000]  510:
[COF]  511: procedure Suche(Name: WortTyp; var gef: boo
       lean);
[000]  512:
[AS9]  513: { Namen in Dictionary suchen }
[000]  514:
[E42]  515: label 99;
[000]  516:
[EMO]  517: var
[D43]  518:   z: integer;
[B53]  519:   n: WortTyp;
[000]  520:
[BH1]  521: begin
[2R2]  522:   z := Anfang;
[032]  523:   n := EMPTY;
[OX2]  524:   gef := false;
[EE4]  525:   while Name <> n do
[BH1]  526:     begin
[R23]  527:       Zeiger := z;
[TS5]  528:       move(M[Zeiger], z, 2);
[B85]  529:       if z = 0 then goto 99;
[PY5]  530:       move(M[Zeiger+2], n, 17);
[TUO]  531:     end;
[T95]  532:   99: gef := Name = n;
[TUO]  533: end;
[000]  534:
[000]  535:
[000]  536:
[87C]  537: function FindeVar(w: worttyp):integer;
[BH1]  538: begin
[FS5]  539:       suche(w, gefunden);
[Y99]  540:       if not(gefunden) then error(4);
[9PE]  541:       if M[Zeiger+length(w)+3] <> Variable
       then error(4);
[ME9]  542:       findevar := Zeiger + length(w) + 12;
[TUO]  543: end;
[000]  544:
[000]  545:
```

```
[000]  546:
[NL5]  547: procedure TueName;
[000]  548:
[LXC]  549: { Name holen und überprüfen, Header bauen }
[000]  550:
[M95]  551: var Name: WortTyp;
[000]  552:
[BH1]  553: begin
[PT4]  554:   Name := HoleWort;
[3C8]  555:   if Name = EMPTY then error(2);
[366]  556:   Suche(Name, gefunden);
[6Y7]  557:   if gefunden then error(3);
[PL2]  558:   os(Name);
[9X4]  559:   main := Name = 'MAIN';
[TU0]  560: end;
[000]  561:
[000]  562:
[000]  563:
[5A6]  564: procedure pushregs;
[BH1]  565: begin
[YE5]  566:   ob($06);    { push es }
[UR5]  567:   ob($1e);    { push ds }
[KE5]  568:   ob($57);    { push di }
[TF5]  569:   ob($56);    { push si }
[WE5]  570:   ob($55);    { push bp }
[TU0]  571: end;
[000]  572:
[000]  573:
[000]  574:
[AZ5]  575: procedure popregs;
[BH1]  576: begin
[PA5]  577:   ob($5d);    { pop bp }
[LB5]  578:   ob($5e);    { pop si }
[CA5]  579:   ob($5f);    { pop di }
[OB5]  580:   ob($1f);    { pop ds }
[425]  581:   ob($07);    { pop es }
[TU0]  582: end;
[000]  583:
[000]  584:
[000]  585:
[RX5]  586: procedure getvars;
[BH1]  587: begin
[7BC]  588:   ob($a1); ow(findevar('AX'));          {
       mov ax,[AX] }
[5FD]  589:   ob($8b); ob($1e); ow(findevar('BX')); {
       mov bx,[BX] }
[7FD]  590:   ob($8b); ob($0e); ow(findevar('CX')); {
       mov cx,[CX] }
[C5D]  591:   ob($8b); ob($16); ow(findevar('DX')); {
       mov dx,[DX] }
[TU0]  592: end;
[000]  593:
[000]  594:
[000]  595:
[VY5]  596: procedure setvars;
[BH1]  597: begin
[PLC]  598:   ob($89); ob($16); ow(findevar('DX')); {
       mov [DX],dx }
[FXC]  599:   ob($89); ob($0e); ow(findevar('CX')); {
       mov [CX],cx }
[BXC]  600:   ob($89); ob($1e); ow(findevar('BX')); {
       mov [BX],bx }
[D1C]  601:   ob($a3); ow(findevar('AX'));          {
       mov [AX],ax }
[TU0]  602: end;
[000]  603:
[000]  604:
[000]  605:
[F1A]  606: procedure TueSystem(w: WortTyp);
[000]  607:
[B98]  608: { SYSTEM-Worte compilieren }
[000]  609:
[E42]  610: label 99;
[000]  611:
[EM0]  612: var
[6K2]  613:   z: char;
[M67]  614:   disp, Fehler, adr: integer;
[000]  615:
[BH1]  616: begin
[5H2]  617:   sys := true;
[000]  618:
[X21]  619:   { ; }
[000]  620:
[044]  621:   if w = ';' then
[BH1]  622:     begin
[HP3]  623:       if main then
[BH1]  624:         begin
[XC6]  625:           ob($B4); ob($00);    { mov ah,0 }
```

```
[T36]  626:           ob($CD); ob($21);    { int 21H  }
[5X4]  627:           ob($90);             { nop      }
[BD2]  628:           addr := pc;
[0G8]  629:           move(addr, M[findevar('TIB')], 2)
       ;
[6A3]  630:           addr := pc + 80;
[4C8]  631:           move(addr, M[findevar('DP')], 2);
[0Z8]  632:           addr := addr + maxallot + $44 + 8
       0 + $80;
[4B8]  633:           move(addr, M[findevar('S0')], 2);
[DL0]  634:         end
[FS6]  635:       else ob($C3);            { ret }
[WZ3]  636:       if sp > 0 then
[F85]  637:         case S[sp-1].typ of
[FB3]  638:           1: error(20);
[LB3]  639:           2: error(21);
[RB3]  640:           3: error(22);
[VB3]  641:           4: error(23);
[0S3]  642:           else error(0);
[TU0]  643:         end;
[DL0]  644:     end
[000]  645:
[411]  646:   { ( }
[000]  647:
[189]  648:   else if w = '(' then TueKommentar
[000]  649:
[UE1]  650:   { [  }
[000]  651:
[5E7]  652:   else if w = '[' then TueCode
[000]  653:
[WH1]  654:   { IF }
[000]  655:
[NK5]  656:   else if w = 'IF' then
[BH1]  657:     begin
[EV5]  658:       if dup then pc := pc - 4;
[AS8]  659:       ob($85); ob($C0);          { test ax,
       ax }
[F0A]  660:       if not dup then ob($AD);   { lodsw
       }
[437]  661:       ob($75); ob(03);           { jnz  +3
       }
[5D8]  662:       ob($e9);                   { jmp   di
       sp  }
[705]  663:       push(pc, 1); ow(0);
[DL0]  664:     end
[000]  665:
[L12]  666:   { ENDIF }
[000]  667:
[WD6]  668:   else if w = 'ENDIF' then
[BH1]  669:     begin
[9T2]  670:       adr := pop(1);
[D27]  671:       M[adr] := lo(near(adr, pc));
[YS7]  672:       M[adr+1] := hi(near(adr, pc));
[DL0]  673:     end
[000]  674:
[CU3]  675:   { ELSE, ENDOF, ;: }
[000]  676:
[YTC]  677:   else if (w = 'ELSE') or (w = 'ENDOF') or
       (w = ';:') then
[BH1]  678:     begin
[9T2]  679:       adr := pop(1);
[PK7]  680:       M[adr] := lo(near(adr, pc+3));
[B08]  681:       M[adr+1] := hi(near(adr, pc+3));
[2U7]  682:       ob($E9);                   { jmp  di
       sp }
[705]  683:       push(pc, 1); ow(0);
[DL0]  684:     end
[000]  685:
[5N1]  686:   { CASE }
[000]  687:
[ZH6]  688:   else if w = 'CASE' then
[BH1]  689:     begin
[708]  690:       if caseflag then error(28);
[VA5]  691:       caseflag:=true;
[DL0]  692:     end
[000]  693:
[EK1]  694:   { OF }
[000]  695:
[TL5]  696:   else if w = 'OF' then
[BH1]  697:     begin
[NR5]  698:       ofcnt := succ(ofcnt);
[718]  699:       ob($2b); ob($04);    { sub ax,[si] }
[SD6]  700:       ob($ad);             { lodsw       }
[L56]  701:       ob($74);ob(03);      { jz  +3      }
[C47]  702:       ob($e9);             { jmp disp    }
[705]  703:       push(pc, 1); ow(0);
[SD6]  704:       ob($ad);             { lodsw       }
[DL0]  705:     end
```

```
[000]  706:
[BB2]  707:    { ENDCASE }
[000]  708:
[X96]  709:    else if w = 'ENDCASE' then
[BH1]  710:      begin
[K06]  711:        ob($ad);              { lodsw }
[3F5]  712:        for i:=1 to ofcnt do
[BH1]  713:          begin
[9T2]  714:            adr := pop(1);
[D27]  715:            M[adr] := lo(near(adr, pc));
[YS7]  716:            M[adr+1] := hi(near(adr, pc));
[TUO]  717:          end;
[2D2]  718:        ofcnt:=0;
[UH5]  719:        caseflag:=false;
[DLO]  720:      end
[000]  721:
[FN1]  722:    { FIND }
[000]  723:
[8M6]  724:    else if w = 'FIND' then
[BH1]  725:      begin
[OW3]  726:        w := HoleWort;
[FS5]  727:        suche(w, gefunden);
[Y99]  728:        if not(gefunden) then error(4);
[E27]  729:        adr := Zeiger + length(w) + 4;
[Y56]  730:        ob($4e);              { dec si      }
[Y56]  731:        ob($4e);              { dec si      }
[RT7]  732:        ob($89); ob($04);     { mov [si],ax }
[ZA9]  733:        ob($b8); ow(adr);     { mov ax,adr  }
[DLO]  734:      end
[000]  735:
[X42]  736:    { INT }
[000]  737:
[346]  738:    else if w = 'INT' then
[BH1]  739:      begin
[079]  740:        ob($a2); ow(pc+19);   { mov [adr],al }
[ES6]  741:        ob($51);              { push cx      }
[VO3]  742:        getvars;
[2T5]  743:        ob($CD);ob(0);        { int ...  }
[713]  744:        setvars;
[BM5]  745:        ob($59);              { pop cx   }
[2Z5]  746:        ob($ad);              { lodsw    }
[DLO]  747:      end
[000]  748:
[US2]  749:    { BEGININT }
[000]  750:
[C86]  751:    else if w = 'BEGININT' then
[BH1]  752:      begin
[RZ2]  753:        pushregs;
[SX5]  754:        ob($0e);    { push cs }
[XM5]  755:        ob($1f);    { pop ds  }
[713]  756:        setvars;
[KXC]  757:        ob($8b); ob($36); ow(findevar('S0'));
         { mov si,[s0] }
[8SC]  758:        ob($89); ob($26); ow(findevar('R0'));
         { mov [r0],sp }
[DLO]  759:      end
[000]  760:
[882]  761:    { ENDINT }
[000]  762:
[PC6]  763:    else if w = 'ENDINT' then
[BH1]  764:      begin
[VO3]  765:        getvars;
[EZ2]  766:        popregs;
[8S4]  767:        ob($cf); { iret }
[DLO]  768:      end
[000]  769:
[ND2]  770:    { EXITTO }
[000]  771:
[4F6]  772:     else if w = 'EXITTO' then
[BH1]  773:       begin
[WC7]  774:         ob($fa);                    { cli
                  }
[KT8]  775:         ob($89);ob($c3);            { mov bx,a
       x     }
[NL9]  776:         ob($8b);ob($07);            { mov ax,[
       bx]   }
[P2A]  777:         ob($a3);ow(pc + 25);        { mov [dis
       p],ax }
[VO3]  778:         getvars;
[EZ2]  779:         popregs;
[7EB]  780:         ob($2e); ob($ff); ob($2e); ow(0); {
       jmp far ... }
[DLO]  781:       end
[000]  782:
[K92]  783:    { CALLTO }
[000]  784:
[F96]  785:     else if w = 'CALLTO' then
[BH1]  786:       begin
```

```
[W27]  787:         ob($fa);                    { cli
                  }
[DM8]  788:         ob($89); ob($c3);           { mov bx,ax
                  }
[TX8]  789:         ob($8b); ob($07);           { mov ax,[bx]
                  }
[4EA]  790:         ob($a3); ow(pc + 26); { mov [disp],a
       x }
[VO3]  791:         getvars;
[EZ2]  792:         popregs;
[2K7]  793:         ob($9c);                    { pushf
                  }
[9LB]  794:         ob($2e); ob($ff); ob($1e); ow(0); {
       call far ... }
[247]  795:         ob($fb);                    { sti
                  }
[RZ2]  796:         pushregs;
[U48]  797:         ob($0e);                    { push cs
                  }
[ZP7]  798:         ob($1f);                    { pop ds
                  }
[X83]  799:         ob($8b); ob($36);
[H5A]  800:         ow(findevar('S0'));         { mov si,[s0]
                  }
[713]  801:         setvars;
[DLO]  802:       end
[000]  803:
[D22]  804:    { BEGIN }
[000]  805:
[VB6]  806:    else if w = 'BEGIN' then
[BH1]  807:      begin
[V23]  808:        push(pc, 2);
[DLO]  809:      end
[000]  810:
[X62]  811:    { UNTIL }
[000]  812:
[8H6]  813:    else if w = 'UNTIL' then
[BH1]  814:      begin
[EV5]  815:        if dup then pc := pc - 4;
[SC8]  816:        ob($85); ob($C0);                 { test ax
       ,ax }
[SOA]  817:        if not dup then ob($AD);    { lodsw
                  }
[DT2]  818:        adr := pop(2);
[WZ6]  819:        ob($75); ob(03);                  { jnz  +3
                  }
[8HB]  820:        ob($e9); ow(near(pc,adr));  { jmp  di
       sp  }
[DLO]  821:      end
[000]  822:
[N32]  823:    { WHILE }
[000]  824:
[ZF6]  825:    else if w = 'WHILE' then
[BH1]  826:      begin
[EV5]  827:        if dup then pc := pc - 4;
[Y49]  828:        ob($85); ob($C0);              { test ax,ax
                  }
[DSA]  829:        if not dup then ob($AD); { lodsw
                  }
[CL7]  830:        ob($75); ob(03);                  { jnz  +3
                  }
[2W7]  831:        ob($e9);                    { jmp  disp
                  }
[BO5]  832:        push(pc, 3); ow(0);
[DLO]  833:      end
[000]  834:
[M62]  835:    { REPEAT }
[000]  836:
[7C6]  837:    else if w = 'REPEAT' then
[BH1]  838:      begin
[HT2]  839:        adr := pop(3);
[PK7]  840:        M[adr] := lo(near(adr, pc+3));
[BO8]  841:        M[adr+1] := hi(near(adr, pc+3));
[DT2]  842:        adr := pop(2);
[43B]  843:        ob($E9); ow(near(pc,adr)); { jmp  dis
       p }
[DLO]  844:      end
[000]  845:
[M12]  846:    { AGAIN }
[000]  847:
[3C6]  848:    else if w = 'AGAIN' then
[BH1]  849:      begin
[DT2]  850:        adr := pop(2);
[R1B]  851:        ob($E9); ow(near(adr,pc)); { jmp  dis
       p }
[DLO]  852:      end
[000]  853:
[HK1]  854:    { DO }
```

```
[000]  855:
[PL5]  856:    else if w = 'DO' then
[BH1]  857:      begin
[KT6]  858:        ob($57);                    { push di
       }
[XS6]  859:        ob($51);                    { push cx
       }
[HP7]  860:        ob($89); ob($C1);           { mov  cx,ax
       }
[7D7]  861:        ob($AD);                    { lodsw
       }
[CR7]  862:        ob($89); ob($C7);           { mov  di,ax
       }
[7D7]  863:        ob($AD);                    { lodsw
       }
[Z23]  864:        push(pc, 4);
[DL0]  865:      end
[000]  866:
[6T1]  867:    { LOOP }
[000]  868:
[2R6]  869:    else if w = 'LOOP' then
[BH1]  870:      begin
[PK6]  871:        ob($41);                        { inc cx
       }
[P68]  872:        ob($39); ob($F9);               { cmp cx,d
       i }
[NT2]  873:        adr := pop(4);
[Y57]  874:        ob($7f); ob(03);                { jg  +3
       }
[BHA]  875:        ob($e9); ow(near(pc,adr));      { jmp adr
       }
[WN6]  876:        ob($59);                        { pop cx
       }
[ZP6]  877:        ob($5F);                        { pop di
       }
[DL0]  878:      end
[000]  879:
[Y32]  880:    { +LOOP }
[000]  881:
[C96]  882:    else if w = '+LOOP' then
[BH1]  883:      begin
[EV5]  884:        if dup then pc := pc - 4;
[BA8]  885:        ob($03); ob($C8);               { add cx,
       ax }
[VV9]  886:        if not dup then ob($AD);        { lodsw
       }
[UA8]  887:        ob($39); ob($F9);               { cmp cx,
       di }
[NT2]  888:        adr := pop(4);
[D07]  889:        ob($7f); ob(03);                { jg  +3
       }
[TKA]  890:        ob($e9); ow(near(pc,adr));      { jmp adr
       }
[6N6]  891:        ob($59);                        { pop cx
       }
[VN6]  892:        ob($5F);                        { pop di
       }
[DL0]  893:      end
[000]  894:
[442]  895:    { -LOOP }
[000]  896:
[S96]  897:    else if w = '-LOOP' then
[BH1]  898:      begin
[EV5]  899:        if dup then pc := pc - 4;
[4E8]  900:        ob($29); ob($C1);               { sub cx,
       ax }
[VV9]  901:        if not dup then ob($AD);        { lodsw
       }
[UA8]  902:        ob($39); ob($F9);               { cmp cx,
       di }
[NT2]  903:        adr := pop(4);
[VZ6]  904:        ob($7c); ob(03);                { jl  +3
       }
[TKA]  905:        ob($e9); ow(near(pc,adr));      { jmp adr
       }
[6N6]  906:        ob($59);                        { pop cx
       }
[VN6]  907:        ob($5F);                        { pop di
       }
[DL0]  908:      end
[000]  909:
[A42]  910:    { /LOOP }
[000]  911:
[4A6]  912:    else if w = '/LOOP' then
[BH1]  913:      begin
[EV5]  914:        if dup then pc := pc - 4;
[NT2]  915:        adr := pop(4);
[838]  916:        ob($03); ob($C8);               { add  cx
       ,ax }
```

```
[SC8]  917:        ob($85); ob($C0);               { test ax
       ,ax }
[SOA]  918:        if not dup then ob($AD);        { lodsw
       }
[GK8]  919:        ob($79); ob($07);               { jns  di
       sp  }
[378]  920:        ob($39); ob($F9);               { cmp  cx
       ,di }
[1L7]  921:        ob($7c); ob($0a);               { jl   +1
       0   }
[DYA]  922:        ob($e9); ow(near(pc,adr));      { jmp  ad
       r   }
[378]  923:        ob($39); ob($F9);               { cmp  cx
       ,di }
[867]  924:        ob($7f); ob(03);                { jg   +3
       }
[DYA]  925:        ob($e9); ow(near(pc,adr));      { jmp  ad
       r   }
[YY6]  926:        ob($59);                        { pop  cx
       }
[5Z6]  927:        ob($5F);                        { pop  di
       }
[DL0]  928:      end
[000]  929:
[LN1]  930:    { (") }
[000]  931:
[9M5]  932:    else if w = '(")' then
[BH1]  933:      begin
[8U1]  934:        adr := pc;
[U41]  935:        ob(0);
[YR4]  936:        z := HoleZeichen;
[128]  937:        while (z <> '"') and (z <> EMPTY) do
[BH1]  938:          begin
[VT7]  939:            ob(ord(z)); z := HoleZeichen;
[AF5]  940:            M[adr] := M[adr] + 1;
[TU0]  941:          end;
[DL0]  942:      end
[000]  943:
[955]  944:    else sys := false;
[TU0]  945: end;
[000]  946:
[000]  947:
[000]  948:
[PR5]  949: procedure TueProc;
[000]  950:
[GND]  951: { Colon-Definition (Prozedur) compilieren }
[000]  952:
[8G2]  953: label ok;
[000]  954:
[EM0]  955: var
[M53]  956:   w: WortTyp;
[GC9]  957:   disp, Fehler, adr, typ: integer;
[000]  958:
[BH1]  959: begin
[AE8]  960:   TueName;          { Header bauen }
[443]  961:   ob(PROZEDUR);
[1LA]  962:   if main then ot(mjmp, pc - mjmp -2);
[000]  963:
[X52]  964:   repeat
[OW3]  965:     w := HoleWort;
[A57]  966:     if w = EMPTY then error(4);
[000]  967:
[VZ5]  968:     { Char-Konstante ? }
[000]  969:
[AD6]  970:     if w[1] = #39 { ' } then
[BH1]  971:       begin
[8G8]  972:         if length(w) <> 3 then error(16);
[977]  973:         if w[3] <> #39 then error(16);
[4D6]  974:         disp := ord(Merker2[2]);
[YZ5]  975:         ob($4E);              { dec si      }
[YZ5]  976:         ob($4E);              { dec si      }
[RT7]  977:         ob($89); ob($04);     { mov [si],ax }
[BA9]  978:         ob($B8); ow(disp);    { mov ax,disp }
[4M2]  979:         goto ok;
[TU0]  980:       end;
[000]  981:
[SU7]  982:     { Numerische Konstante ? }
[000]  983:
[SZG]  984:     if (length(w) = 1) and (w[1] in ['+', '
       -', '0', '1', '2']) then Fehler := 1
[2N7]  985:     else val(w, disp, Fehler);
[GG4]  986:     if Fehler = 0 then
[BH1]  987:       begin
[YZ5]  988:         ob($4E);              { dec si      }
[YZ5]  989:         ob($4E);              { dec si      }
[RT7]  990:         ob($89); ob($04);     { mov [si],ax }
[BA9]  991:         ob($B8); ow(disp);    { mov ax,disp }
```

```
[4M2]  992:          goto ok;
[TUO]  993:        end;
[000]  994:
[W35]  995:      { System-Wort ? }
[000]  996:
[1Z3]  997:      TueSystem(w);
[OU5]  998:      if sys then goto ok;
[000]  999:
[3M7] 1000:      { in Dictionary suchen }
[000] 1001:
[FS5] 1002:      suche(w, gefunden);
[Y99] 1003:      if not(gefunden) then error(4);
[E27] 1004:      adr := Zeiger + length(w) + 4;
[9U3] 1005:      typ := M[adr-1];
[000] 1006:
[873] 1007:      { Makro ? }
[000] 1008:
[VN4] 1009:      if typ = MAKRO then
[BH1] 1010:        begin
[ML6] 1011:          move(M[Zeiger], disp, 2);
[W95] 1012:          while adr < disp do
[BH1] 1013:            begin
[8S2] 1014:              ob(M[adr]);
[613] 1015:              adr := adr + 1;
[TUO] 1016:            end;
[4M2] 1017:          goto ok;
[TUO] 1018:        end;
[000] 1019:
[A9E] 1020:      { sonst Konstante, Variable oder Prozed
          ur }
[000] 1021:
[LL4] 1022:      disp := adr - pc - 3;
[MK8] 1023:      ob($E8); ow(disp); { call disp }
[R18] 1024:      if w = '."' then TueSystem('(")');
[T88] 1025:      if w = '"' then TueSystem('(")');
[LFO] 1026:      ok:
[HL3] 1027:    until w = ';';
[SG3] 1028:    ot(Ende, pc);
[FB2] 1029:    Ende := pc;
[461] 1030:    ow(0);
[TUO] 1031:  end;
[000] 1032:
[000] 1033:
[000] 1034:
[NW5] 1035:  procedure TueMacro;
[000] 1036:
[OUA] 1037:  { Makro-Definition compilieren }
[000] 1038:
[8G2] 1039:  label ok;
[000] 1040:
[EMO] 1041:  var
[M53] 1042:    w: WortTyp;
[GC9] 1043:    disp, Fehler, adr, typ: integer;
[000] 1044:
[BH1] 1045:  begin
[2M7] 1046:    TueName;     { Header bauen }
[362] 1047:    ob(MAKRO);
[000] 1048:
[X52] 1049:    repeat
[OW3] 1050:      w := HoleWort;
[A57] 1051:      if w = EMPTY then error(4);
[000] 1052:
[VZ5] 1053:      { Char-Konstante ? }
[000] 1054:
[AD6] 1055:      if w[1] = #39 { ' } then
[BH1] 1056:        begin
[8G8] 1057:          if length(w) <> 3 then error(16);
[977] 1058:          if w[3] <> #39 then error(16);
[4D6] 1059:          disp := ord(Merker2[2]);
[YZ5] 1060:          ob($4E);              { dec si      }
[YZ5] 1061:          ob($4E);              { dec si      }
[RT7] 1062:          ob($89); ob($04);     { mov [si],ax }
[BA9] 1063:          ob($B8); ow(disp);    { mov ax,disp }
[4M2] 1064:          goto ok;
[TUO] 1065:        end;
[000] 1066:
[SU7] 1067:      { Numerische Konstante ? }
[000] 1068:
[405] 1069:      val(w, disp, Fehler);
[GG4] 1070:      if Fehler = 0 then
[BH1] 1071:        begin
[YZ5] 1072:          ob($4E);              { dec si      }
[YZ5] 1073:          ob($4E);              { dec si      }
[RT7] 1074:          ob($89); ob($04);     { mov [si],ax }
[BA9] 1075:          ob($B8); ow(disp);    { mov ax,disp }
[4M2] 1076:          goto ok;
[TUO] 1077:        end;
```

```
[000] 1078:
[8RB] 1079:      if w = ';' then goto ok  { Makro-Ende }
[189] 1080:      else if w = '(' then TueKommentar
[5E7] 1081:      else if w = '[' then TueCode
[A11] 1082:      else
[VCB] 1083:        begin { Wort in Dictionary suchen }
[FS5] 1084:          suche(w, gefunden);
[Y99] 1085:          if not(gefunden) then error(4);
[E27] 1086:          adr := Zeiger + length(w) + 4;
[9U3] 1087:          typ := M[adr-1];
[VN4] 1088:          if typ = MAKRO then
[BH1] 1089:            begin
[ML6] 1090:              move(M[Zeiger], disp, 2);
[W95] 1091:              while adr < disp do
[BH1] 1092:                begin
[8S2] 1093:                  ob(M[adr]);
[613] 1094:                  adr := adr + 1;
[TUO] 1095:                end;
[DLO] 1096:            end
[SYB] 1097:          else error(5); { Konstanten, Variab
          le und }
[7KB] 1098:        end;              { Prozeduren nicht e
          rlaubt }
[LFO] 1099:      ok:
[HL3] 1100:    until w = ';';
[SG3] 1101:    ot(Ende, pc);
[FB2] 1102:    Ende := pc;
[461] 1103:    ow(0);
[TUO] 1104:  end;
[000] 1105:
[000] 1106:
[000] 1107:
[697] 1108:  procedure TueVariable;
[000] 1109:
[EAB] 1110:  { Variablen-Definition compilieren }
[000] 1111:
[EMO] 1112:  var
[M53] 1113:    w: WortTyp;
[OH5] 1114:    n, Fehler: integer;
[000] 1115:
[BH1] 1116:  begin
[AE8] 1117:    TueName;          { Header bauen }
[7X2] 1118:    ob(VARIABLE);
[OW3] 1119:    w := HoleWort;
[D57] 1120:    if w = EMPTY then error(7);
[000] 1121:
[VZ5] 1122:    { Char-Konstante ? }
[000] 1123:
[AD6] 1124:    if w[1] = #39 { ' } then
[BH1] 1125:      begin
[8G8] 1126:        if length(w) <> 3 then error(16);
[977] 1127:        if w[3] <> #39 then error(16);
[NW5] 1128:        n := ord(Merker2[2]);
[DLO] 1129:      end
[2L9] 1130:    else      { numerische Konstante }
[BH1] 1131:      begin
[VG4] 1132:        val(w, n, Fehler);
[T77] 1133:        if Fehler <> 0 then error(7);
[TUO] 1134:      end;
[YZ5] 1135:    ob($4E);              { dec si      }
[YZ5] 1136:    ob($4E);              { dec si      }
[RT7] 1137:    ob($89); ob($04);     { mov [si],ax }
[2X7] 1138:    ob($B8); ow(pc+3);    { mov ax,n    }
[GN5] 1139:    ob($C3);              { ret         }
[WD1] 1140:    ow(n);
[SG3] 1141:    ot(Ende, pc);
[FB2] 1142:    Ende := pc;
[461] 1143:    ow(0);
[TUO] 1144:  end;
[000] 1145:
[000] 1146:
[000] 1147:
[Z28] 1148:  procedure TueKonstante;
[000] 1149:
[1GC] 1150:  { Konstanten-Definition compilieren }
[000] 1151:
[EMO] 1152:  var
[M53] 1153:    w: WortTyp;
[OH5] 1154:    n, Fehler: integer;
[000] 1155:
[BH1] 1156:  begin
[EV8] 1157:    TueName;          { Header bauen }
[RC3] 1158:    ob(KONSTANTE);
[OW3] 1159:    w := HoleWort;
[D57] 1160:    if w = EMPTY then error(7);
[000] 1161:
[VZ5] 1162:    { Char-Konstante ? }
```

```
[000] 1163:
[AD6] 1164:    if w[1] = #39 { ' } then
[BH1] 1165:      begin
[8G8] 1166:        if length(w) <> 3 then error(16);
[977] 1167:        if w[3] <> #39 then error(16);
[NW5] 1168:        n := ord(Merker2[2]);
[DL0] 1169:      end
[2L9] 1170:    else     { numerische Konstante }
[BH1] 1171:      begin
[VG4] 1172:        val(w, n, Fehler);
[T77] 1173:        if Fehler <> 0 then error(7);
[TU0] 1174:      end;
[PX5] 1175:    ob($4E);           { dec si      }
[PX5] 1176:    ob($4E);           { dec si      }
[488] 1177:    ob($89); ob($04); { mov [si],ax }
[OK7] 1178:    ob($B8); ow(n);    { mov ax,n    }
[DV5] 1179:    ob($C3);           { ret         }
[SG3] 1180:    ot(Ende, pc);
[FB2] 1181:    Ende := pc;
[461] 1182:    ow(0);
[TU0] 1183: end;
[000] 1184:
[000] 1185:
[000] 1186:
[7L5] 1187: procedure TueVoc;
[000] 1188:
[5H6] 1189: { Vocabulary merken }
[000] 1190:
[BH1] 1191: begin
[G06] 1192:    VocName := HoleWort;
[C19] 1193:    if VocName = EMPTY then error(2);
[PR4] 1194:    VocAnfang := Ende;
[TU0] 1195: end;
[000] 1196:
[000] 1197:
[000] 1198:
[6T6] 1199: procedure SichereVoc;
[000] 1200:
[88A] 1201: { aktuelles Vocabulary sichern }
[000] 1202:
[EM0] 1203: var
[UH2] 1204:    v: file;
[W33] 1205:    i: integer;
[ZY2] 1206:    b1, b2: byte;
[000] 1207:
[BH1] 1208: begin
[128] 1209:    if VocAnfang = 0 then error(24);
[PU6] 1210:    assign(v, VocName + '.VOC');
[XS3] 1211:    rewrite(v,1);
[9Z2] 1212:    writeln;
[9MA] 1213:    writeln('Sichern ', VocName, '.VOC ...');
[4Z4] 1214:    b1 := lo(VocAnfang);
[AY4] 1215:    b2 := hi(VocAnfang);
[ME5] 1216:    BlockWrite(v, b1, 1);
[ME5] 1217:    BlockWrite(v, b2, 1);
[VBC] 1218:    BlockWrite(v, M[VocAnfang], pc - VocAnfan
      g);
[KN2] 1219:    close(v);
[XX5] 1220:    writeln('Fertig');
[9Z2] 1221:    writeln;
[TU0] 1222: end;
[000] 1223:
[000] 1224:
[000] 1225:
[EV5] 1226: procedure Include;
[000] 1227:
[WG7] 1228: { Vocabulary einbinden }
[000] 1229:
[EM0] 1230: var
[UZ3] 1231:    Name: WortTyp;
[UH2] 1232:    v: file;
[W33] 1233:    i: integer;
[ZY2] 1234:    b1, b2: byte;
[000] 1235:
[BH1] 1236: begin
[PT4] 1237:    Name := HoleWort;
[3C8] 1238:    if Name = EMPTY then error(2);
[RR4] 1239:    Name := Name + '.VOC';
[9Z2] 1240:    writeln;
[NH8] 1241:    writeln('Lesen ', Name, ' ...');
[BL4] 1242:    assign(v, Name);
[X66] 1243:    {$I-} reset(v,1); {$I+}
[P38] 1244:    if IOresult <> 0 then error(25);
[BX4] 1245:    BlockRead(v, b1, 1);
[KX4] 1246:    BlockRead(v, b2, 1);
[PG3] 1247:    i := b2 shl 8 + b1;
[ON7] 1248:    if i <> Ende then error(26);
[8K2] 1249:    pc := pc - 2;
```

```
[LR9] 1250:    BlockRead(v, M[pc], FileSize(v) - 2);
[AA6] 1251:    pc := pc + FileSize(v) - 2;
[V93] 1252:    Ende := pc - 2;
[KN2] 1253:    close(v);
[XX5] 1254:    writeln('Fertig');
[9Z2] 1255:    writeln;
[TU0] 1256: end;
[000] 1257:
[000] 1258:
[6G9] 1259: procedure compile(w: WortTyp);
[000] 1260:
[N36] 1261: { Wort compilieren }
[000] 1262:
[LP2] 1263: var i: byte;
[000] 1264:
[BH1] 1265: begin
[NX6] 1266:    if w = ':' then TueProc
[K29] 1267:    else if w = 'MACRO' then TueMacro
[4YA] 1268:    else if w = 'VARIABLE' then TueVariable
[M8B] 1269:    else if w = 'CONSTANT' then TueKonstante
[189] 1270:    else if w = '(' then TueKommentar
[CL9] 1271:    else if w = 'VOCABULARY' then TueVoc
[RV8] 1272:    else if w = 'INCLUDE' then Include
[A18] 1273:    else if w = '(DUMP)' then dump
[BEB] 1274:    else if w = '(LIST.ON)' then ListFlag :=
       true
[ZTC] 1275:    else if w = '(LIST.OFF)' then ListFlag :=
       false
[XSA] 1276:    else if w = '(SAVE.VOC)' then SichereVoc
[4S3] 1277:    else error(1);
[TU0] 1278: end;
[000] 1279:
[000] 1280:
[000] 1281:
[9L8] 1282: { Hauptprogramm S-Forth }
[000] 1283:
[000] 1284:
[BH1] 1285: begin
[NF2] 1286:    ClrScr;
[9Z2] 1287:    writeln;
[XAD] 1288:    writeln('*** S-Forth Compiler Vers. ', VE
      RSION, ' ***');
[9Z2] 1289:    writeln;
[FW9] 1290:    writeln('/// DOS International');
[9Z2] 1291:    writeln;
[RB1] 1292:    init;
[Y59] 1293:    writeln('Start Compilation');
[9Z2] 1294:    writeln;
[BH1] 1295:    begin
[X23] 1296:      ob($8B); ob($36);
[2Z2] 1297:      sinit := pc;
[N18] 1298:      ow(0);                    { mov si,[SO
      ]    }
[8Z8] 1299:      ob($33); ob($C0);         { xor ax,ax
       }
[1T8] 1300:      ob($E9); ow($0000);       { jmp main
       }
[TU0] 1301:    end;
[3E3] 1302:    Anfang := pc;
[FB2] 1303:    Ende := pc;
[461] 1304:    ow(0);
[3X2] 1305:    mjmp := pc-4;
[6X4] 1306:    Wort := HoleWort;
[CZ9] 1307:    while (Wort <> EMPTY) and (not main) do
[BH1] 1308:      begin
[AH4] 1309:        compile(Wort);
[6X4] 1310:        Wort := HoleWort;
[TU0] 1311:      end;
[637] 1312:    ot(sinit, findevar('SO'));
[S87] 1313:    if NoCodeFlag = false then
[BH1] 1314:      begin
[NN7] 1315:        if not main then error(27);
[2Y5] 1316:        writeln; writeln;
[HB9] 1317:        writeln('Speichern ', Name + '.com');
[CC8] 1318:        assign(outfile, Name + '.com');
[OL7] 1319:        rewrite(outfile, pc-256);
[6T8] 1320:        BlockWrite(outfile, M[256], 1);
[XR4] 1321:        close(outfile);
[TU0] 1322:      end;
[XX5] 1323:    writeln('Fertig');
[5T0] 1324: end.
```

Listing 1. Der Kern des SForth-Compilers in Turbo Pascal 3.0

```
[4ND]    0: ( *** S-Forth -- System Vocabulary Version
           1.5 *** )
[000]    1:
[000]    2:
[000]    3:
[SP6]    4: ( ---------------------------------------
           )
[2XB]    5: ( *** SForth ist ein Software-Projekt von
           )
[K2D]    6: ( DOS International. Beschreibung siehe DOS
           )
[LLA]    7: ( Ausgabe 9/87 ff und DOS Extra 2, 1988
           )
[299]    8: ( [c] 1987 Volker Everts
           )
[5WA]    9: ( Redaktion DOS International
           )
[S69]   10: ( Elfenstr. 40, 8000 München 83
           )
[SP6]   11: ( ---------------------------------------
           )
[000]   12:
[000]   13:
[000]   14:
[KLC]   15: vocabulary forth     ( System-Vocabulary )
[000]   16:
[000]   17:
[000]   18:
[ZG6]   19: ( *** System-Variablen *** )
[000]   20:
[DK8]   21: variable r0 $0000    ( Return-Stack )
[1B8]   22: variable s0 $F800    ( Daten-Stack )
[H6A]   23: variable dp $4000    ( freier Speicher )
[299]   24: variable base 10     ( Zahlenbasis )
[A4B]   25: variable hld 0       ( Zeiger auf Ziffer )
[S6D]   26: variable dpl 0       ( Zeiger auf Dezimalpun
           kt )
[9YC]   27: variable out 0       ( Zeichenzähler für EMI
           T )
[SZB]   28: variable in 0        ( Zeichenzähler Eingabe
           )
[3SB]   29: variable tib $F000   ( Terminal Input Buffer
           )
[P7H]   30: variable leading '0' ( enthält führende Nul
           l oder Space )
[NF9]   31: variable adr 0       ( String-Adresse )
[Z7B]   32: variable ml 0        ( maximale String-Länge
           )
[000]   33:
[000]   34:
[000]   35:
[KB7]   36: ( *** System-Konstanten *** )
[000]   37:
[7YE]   38: constant dictionary $108  ( Start Dictionar
           y-Link )
[TD6]   39: constant 0 0      ( Null )
[WC6]   40: constant 1 1      ( Eins )
[0E6]   41: constant 2 2      ( Zwei )
[S39]   42: constant bl 32    ( Leerzeichen )
[TY7]   43: constant bs 8     ( Backspace )
[RU8]   44: constant true -1  ( True-Flag )
[WF9]   45: constant false 0  ( False-Flag )
[000]   46:
[000]   47:
[000]   48:
[ZK7]   49: ( *** Stack-Manipulation *** )
[000]   50:
[FV9]   51: macro dup ( n --> n n ) ( TOS kopieren )
[UN4]   52:    [ $4E ]      ( dec si       )
[UN4]   53:    [ $4E ]      ( dec si       )
[3C5]   54:    [ $89 $04 ]  ( mov [si],ax  )
[V10]   55: ;
[000]   56:
[000]   57:
[359]   58: macro drop ( n --> ) ( TOS entfernen )
[2H4]   59:    [ $AD ] ( lodsw )
[V10]   60: ;
[000]   61:
[000]   62:
[12F]   63: macro swap ( n1 n2 --> n2 n1 ) ( TOS und SE
           COND vertauschen )
[FN5]   64:    [ $87 $04 ] ( xchg ax,[si] )
[V10]   65: ;
[000]   66:
[000]   67:
[6ND]   68: macro >r ( n --> ) ( TOS auf Return-Stack s
           peichern )
[D46]   69:    [ $50 ] ( push ax ) drop
[V10]   70: ;
```

```
[000]   71:
[000]   72:
[EOD]   73: macro r> ( --> n ) ( TOS von Return-Stack h
           olen )
[HM4]   74:    dup [ $58 ] ( pop ax )
[V10]   75: ;
[000]   76:
[000]   77:
[Z9E]   78: macro r ( --> n ) ( TOS von Return-Stack ko
           pieren )
[LT4]   79:    r> [ $50 ] ( push ax )
[V10]   80: ;
[000]   81:
[000]   82:
[KKB]   83: : over ( n1 n2 --> n1 n2 n1 ) ( SECOND kopi
           eren )
[UL7]   84:    dup [ $8B $44 $02 ] ( mov ax,[si+2] )
[V10]   85: ;
[000]   86:
[000]   87:
[BWE]   88: : -dup ( n --> n ? ) ( TOS kopieren, wenn u
           ngleich Null )
[NS4]   89:    dup if dup endif
[V10]   90: ;
[000]   91:
[000]   92:
[T2F]   93: : rot ( n1 n2 n3 --> n2 n3 n1 ) ( zyklische
           s Vertauschen )
[UD5]   94:    >r swap r> swap
[V10]   95: ;
[000]   96:
[000]   97:
[1AD]   98: : ddup ( d --> d d ) ( doppeltgenaue Zahl k
           opieren )
[ST2]   99:    over over
[V10]  100: ;
[000]  101:
[000]  102:
[000]  103:
[ZV7]  104: ( *** einfache Arithmetik *** )
[000]  105:
[6D8]  106: macro + ( n1 n2 --> n ) ( Addition )
[RN5]  107:    [ $01 $04 ] ( add  [si],ax )
[N21]  108:    drop
[V10]  109: ;
[000]  110:
[000]  111:
[S2A]  112: macro - ( n1 n2 --> n ) ( Subtraktion )
[3V5]  113:    [ $29 $04 ] ( sub  [si],ax )
[N21]  114:    drop
[V10]  115: ;
[000]  116:
[000]  117:
[ROB]  118: macro * ( n1 n2 --> p ) ( Multiplikation )
[4N5]  119:    [ $F7 $2C ] ( imul ax,[si] )
[H56]  120:    [ $46 $46 ] ( inc si; inc si )
[V10]  121: ;
[000]  122:
[000]  123:
[SN6]  124: : / ( n1 n2 --> q ) ( Division )
[CW4]  125:    [ $51 ]      ( push cx     )
[635]  126:    [ $89 $C1 ]  ( mov  cx,ax  )
[7K4]  127:    [ $AD ]      ( lodsw       )
[R45]  128:    [ $33 $D2 ]  ( xor  dx,dx  )
[GS4]  129:    [ $F7 $F9 ]  ( idiv cx     )
[1T4]  130:    [ $59 ]      ( pop  cx     )
[V10]  131: ;
[000]  132:
[000]  133:
[EL9]  134: : mod ( n1 n2 --> r ) ( Divisionsrest )
[1A5]  135:    / [ $89 $D0 ] ( mov ax,dx )
[V10]  136: ;
[000]  137:
[000]  138:
[5VE]  139: : /mod ( n1 n2 --> r q ) ( Divisonsrest und
           Quotient )
[F10]  140:    /
[UN4]  141:    [ $4E ]      ( dec si       )
[UN4]  142:    [ $4E ]      ( dec si       )
[3C5]  143:    [ $89 $14 ]  ( mov [si],dx  )
[V10]  144: ;
[000]  145:
[000]  146:
[RT9]  147: : */ ( n1 n2 n3 --> n ) ( berechnet n1*n2/n
           3 )
[Y35]  148:    [ $51 ]      ( push cx       )
[SA5]  149:    [ $89 $C1 ]  ( mov  cx,ax    )
[TT4]  150:    [ $AD ]      ( lodsw         )
```

```
[4N5]  151:    [ $F7 $2C ] ( imul ax,[si] )
[205]  152:    [ $F7 $F9 ] ( idiv cx      )
[L05]  153:    [ $59 ]     ( pop  cx      )
[V10]  154: ;
[000]  155:
[000]  156:
[73E]  157: : */mod ( n1 n2 n3 --> r n ) ( berechnet Re
       st und n1*n2/n3 )
[840]  158:    */
[UN4]  159:    [ $4E ]     ( dec si       )
[UN4]  160:    [ $4E ]     ( dec si       )
[3C5]  161:    [ $89 $14 ] ( mov [si],dx )
[V10]  162: ;
[000]  163:
[000]  164:
[6KB]  165: macro and ( n1 n2 --> n ) ( And-Verknüpfung
       )
[X65]  166:    [ $21 $04 ] ( and [si],ax )
[N21]  167:    drop
[V10]  168: ;
[000]  169:
[000]  170:
[D4B]  171: macro or ( n1 n2 --> n ) ( Or-Verknüpfung )
[185]  172:    [ $09 $04 ] ( or [si],ax )
[N21]  173:    drop
[V10]  174: ;
[000]  175:
[000]  176:
[GYB]  177: macro xor ( n1 n2 --> n ) ( Xor-Verknüpfung
       )
[GC5]  178:    [ $31 $04 ] ( xor [si],ax )
[N21]  179:    drop
[V10]  180: ;
[000]  181:
[000]  182:
[XWC]  183: macro minus ( n --> -n ) ( Vorzeichenwechse
       l )
[9K4]  184:    [ $F7 $D8 ] ( neg  ax )
[V10]  185: ;
[000]  186:
[000]  187:
[WN9]  188: macro not ( n --> n1 ) ( Not-Funktion )
[EK4]  189:    [ $F7 $D0 ] ( not  ax )
[V10]  190: ;
[000]  191:
[000]  192:
[KS9]  193: macro 1+ ( n --> n1 ) ( TOS um 1 erhöhen )
[HS3]  194:    [ $40 ]  ( inc ax )
[V10]  195: ;
[000]  196:
[000]  197:
[TS9]  198: macro 2+ ( n --> n1 ) ( TOS um 2 erhöhen )
[2M0]  199:    1+ 1+
[V10]  200: ;
[000]  201:
[000]  202:
[WVA]  203: macro 1- ( n --> n1 ) ( TOS um 1 erniedrige
       n )
[SR3]  204:    [ $48 ]  ( dec ax )
[V10]  205: ;
[000]  206:
[000]  207:
[3WA]  208: macro 2- ( n --> n1 ) ( TOS um 2 erniedrige
       n )
[GM0]  209:    1- 1-
[V10]  210: ;
[000]  211:
[000]  212:
[000]  213:
[6KB]  214: ( *** Double, Unsigned, Mixed Arithmetik **
       * )
[000]  215:
[000]  216:
[7L9]  217: : d+ ( d1 d2 --> d ) ( doppeltgenaue Summe
       )
[A36]  218:    [ $8B $14    ] ( mov dx,[si]    )
[RS6]  219:    [ $01 $54 4  ] ( add [si+4],dx  )
[4S6]  220:    [ $11 $44 2  ] ( adc [si+2],ax  )
[VP2]  221:    drop drop
[V10]  222: ;
[000]  223:
[000]  224:
[MUG]  225: : u* ( u1 u2 --> ud ) ( doppeltgenaues Prod
       ukt ohne Vorzeichen )
[6L5]  226:    [ $F7 $24 ] ( mul ax,[si] )
[3C5]  227:    [ $89 $04 ] ( mov [si],ax )
[UK5]  228:    [ $89 $D0 ] ( mov ax,dx   )
[V10]  229: ;
[000]  230:
[000]  231:
```

```
[H4N]  232: : u/ ( ud u --> Rest Quotient ) ( doppeltge
       naue Division ohne Vorzeichen )
[CW4]  233:    [ $51 ]     ( push cx     )
[635]  234:    [ $89 $C1 ] ( mov  cx,ax  )
[SL5]  235:    [ $8B $14 ] ( mov dx,[si] )
[GN4]  236:    [ $AD ]     ( lodsw       )
[GN4]  237:    [ $AD ]     ( lodsw       )
[MG4]  238:    [ $F7 $F1 ] ( div cx      )
[AW4]  239:    [ $59 ]     ( pop  cx     )
[UN4]  240:    [ $4E ]     ( dec si      )
[UN4]  241:    [ $4E ]     ( dec si      )
[3C5]  242:    [ $89 $14 ] ( mov [si],dx )
[V10]  243: ;
[000]  244:
[000]  245:
[7SF]  246: : m* ( n1 n2 --> d ) ( doppeltgenaues Produ
       kt mit Vorzeichen )
[4N5]  247:    [ $F7 $2C ] ( imul ax,[si] )
[5V5]  248:    [ $89 $04 ] ( mov  [si],ax )
[4B5]  249:    [ $89 $D0 ] ( mov  ax,dx   )
[V10]  250: ;
[000]  251:
[000]  252:
[WPF]  253: : m/ ( d n --> r q ) ( doppeltgenaue Divisi
       on mit Vorzeichen )
[NZ4]  254:    [ $51 ]     ( push cx     )
[F65]  255:    [ $89 $C1 ] ( mov  cx,ax  )
[SL5]  256:    [ $8B $14 ] ( mov dx,[si] )
[GN4]  257:    [ $AD ]     ( lodsw       )
[GN4]  258:    [ $AD ]     ( lodsw       )
[TV4]  259:    [ $F7 $F9 ] ( idiv cx     )
[AW4]  260:    [ $59 ]     ( pop  cx     )
[UN4]  261:    [ $4E ]     ( dec si      )
[UN4]  262:    [ $4E ]     ( dec si      )
[3C5]  263:    [ $89 $14 ] ( mov [si],dx )
[V10]  264: ;
[000]  265:
[000]  266:
[H6K]  267: : m/mod ( ud u --> r dq ) ( Rest und doppel
       tgenauer Quotient )
[029]  268:    dup >r 0 swap u/ rot rot r> u/ rot
[V10]  269: ;
[000]  270:
[000]  271:
[000]  272:
[1H6]  273: ( *** Speicher-Befehle *** )
[000]  274:
[AB8]  275: macro @ ( adr --> n ) ( Zahl holen )
[EK5]  276:    [ $89 $C3 ] ( mov bx,ax   )
[VF5]  277:    [ $8B $07 ] ( mov ax,[bx] )
[V10]  278: ;
[000]  279:
[000]  280:
[DG7]  281: : c@ ( adr --> c ) ( Zeichen holen )
[PA5]  282:    [ $89 $C3 ] ( mov  bx,ax   )
[AF5]  283:    [ $8A $07 ] ( mov  al,[bx] )
[RX4]  284:    [ $B4 $00 ] ( mov  ah,0    )
[V10]  285: ;
[000]  286:
[000]  287:
[VH8]  288: : ! ( n adr --> ) ( Zahl speichern )
[PA5]  289:    [ $89 $C3 ] ( mov  bx,ax   )
[N21]  290:    drop
[DV5]  291:    [ $89 $07 ] ( mov  [bx],ax )
[N21]  292:    drop
[V10]  293: ;
[000]  294:
[000]  295:
[CL9]  296: : c! ( c adr --> ) ( Zeichen speichern )
[PA5]  297:    [ $89 $C3 ] ( mov  bx,ax   )
[N21]  298:    drop
[GU5]  299:    [ $88 $07 ] ( mov  [bx],al )
[N21]  300:    drop
[V10]  301: ;
[000]  302:
[000]  303:
[PFB]  304: : +! ( n adr --> ) ( Zahl im Speicher addie
       ren )
[PA5]  305:    [ $89 $C3 ] ( mov  bx,ax   )
[N21]  306:    drop
[ZN5]  307:    [ $01 $07 ] ( add  [bx],ax )
[N21]  308:    drop
[V10]  309: ;
[000]  310:
[000]  311:
[8GB]  312: : ++ ( adr --> ) ( Speicherwort um 1 erhöhe
       n )
[435]  313:    [ $89 $C3 ] ( mov  bx,ax )
[N21]  314:    drop
[ME5]  315:    [ $FF $07 ] ( inc  [bx]  )
```

```
[V10]  316: ;
[000]  317:
[000]  318:
[8OD]  319: : -- ( adr --> ) ( Speicherwort um 1 ernied
           rigen )
[435]  320:   [ $89 $C3 ] ( mov  bx,ax )
[N21]  321:   drop
[OE5]  322:   [ $FF $OF ] ( dec  [bx]  )
[V10]  323: ;
[000]  324:
[000]  325:
[U2H]  326: : cmove ( von bis n --> ) ( Speicherbereich
           verschieben )
[4E5]  327:   [ $51 ]       ( push cx     )
[W75]  328:   [ $89 $C1 ]   ( mov  cx,ax  )
[N21]  329:   drop
[ZB5]  330:   [ $57 ]       ( push di     )
[Z45]  331:   [ $89 $C7 ]   ( mov  di,ax  )
[N21]  332:   drop
[ME5]  333:   [ $56 ]       ( push si     )
[S75]  334:   [ $89 $C6 ]   ( mov  si,ax  )
[Z25]  335:   [ $39 $F7 ]   ( cmp  di,si  )
[B44]  336:   [ $7C 7   ]   ( jl   +7     )
[Z25]  337:   [ $03 $F1 ]   ( add  si,cx  )
[DO5]  338:   [ $03 $F9 ]   ( add  di,cx  )
[Z45]  339:   [ $4E ]       ( dec  si     )
[A25]  340:   [ $4F ]       ( dec  di     )
[194]  341:   [ $FD ]       ( std         )
[V65]  342:   [ $F3 $A4 ]   ( rep  movsb  )
[W74]  343:   [ $FC ]       ( cld         )
[A75]  344:   [ $5E ]       ( pop  si     )
[N45]  345:   [ $5F ]       ( pop  di     )
[R55]  346:   [ $59 ]       ( pop  cx     )
[N21]  347:   drop
[V10]  348: ;
[000]  349:
[000]  350:
[58H]  351: : fill ( adr n c --> ) ( Speicherbereich mi
           t Zeichen c füllen )
[ZS5]  352:   [ $89 $C2 ]   ( mov  dx,ax      )
[N21]  353:   drop
[OZ5]  354:   [ $51 ]       ( push cx         )
[SS5]  355:   [ $89 $C1 ]   ( mov  cx,ax      )
[N21]  356:   drop
[VW5]  357:   [ $57 ]       ( push di         )
[VN5]  358:   [ $89 $C7 ]   ( mov  di,ax      )
[ES5]  359:   [ $89 $DO ]   ( mov  ax,dx      )
[BU5]  360:   [ $F3 $AA ]   ( rep  stosb      )
[HN5]  361:   [ $5F ]       ( pop  di         )
[LP5]  362:   [ $59 ]       ( pop  cx         )
[N21]  363:   drop
[V10]  364: ;
[000]  365:
[000]  366:
[GPF]  367: : erase O fill ; ( Speicherbereich mit Null
           füllen )
[000]  368:
[000]  369:
[NSK]  370: : blanks bl fill ; ( Speicherbereich mit Le
           erzeichen füllen )
[000]  371:
[000]  372:
[V8M]  373: : toggle over c@ xor swap c! ; ( Xor-Verknü
           pfung Speicherstelle )
[000]  374:
[000]  375:
[SPD]  376: : here dp @ ; ( Nächste freie Stelle auf de
           m Heap )
[000]  377:
[000]  378:
[VYB]  379: : pad here $44 + ; ( Adresse des PAD-Bereic
           hes )
[000]  380:
[000]  381:
[S6K]  382: : allot dp +! ; ( n --> ) ( Speicherplatz a
           uf dem Heap allokieren )
[000]  383:
[000]  384:
[CGE]  385: : , here ! 2 allot ; ( Ein Speicherwort all
           okieren )
[000]  386:
[000]  387:
[XGH]  388: : c, here c! 1 allot ; ( Ein Zeichen Speich
           erplatz allokieren )
[000]  389:
[000]  390:
[RZA]  391: macro sp@ ( Datenstackpointer holen )
[YMO]  392:   dup
[TC5]  393:   [ $89 $FO ] ( mov ax,si )
[V10]  394: ;
[000]  395:
```

```
[000]  396:
[XHB]  397: macro sp! ( Datenstackpointer setzen )
[PC5]  398:   [ $89 $C6 ] ( mov si,ax )
[DD5]  399:   [ $33 $CO ] ( xor ax,ax )
[V10]  400: ;
[000]  401:
[000]  402:
[4GB]  403: macro rp@ ( Returnstackpointer holen )
[YMO]  404:   dup
[AE5]  405:   [ $89 $EO ] ( mov ax,sp )
[V10]  406: ;
[000]  407:
[000]  408:
[N3C]  409: macro rp! ( Returnstackpointer setzen )
[GD5]  410:   [ $89 $C4 ] ( mov sp,ax )
[N21]  411:   drop
[V10]  412: ;
[000]  413:
[000]  414:
[5SF]  415: : pick ( n1 n2...ni nj nk... nn j --> n1 n2
           ...ni nj nk...nn nj )
[BVB]  416:   dup + sp@ + @    ( Wert aus Datenstack ho
           len )
[V10]  417: ;
[000]  418:
[000]  419:
[PEE]  420: : roll ( n1 n2...ni nj nk...nn j --> n1 n2.
           ..ni nk...nn nj )
[C5D]  421:   dup >r pick       ( Stackelement herauspi
           cken )
[R5E]  422:   dup sp@ dup 2+    ( Stackpointer-Adresse
           holen )
[FYB]  423:   r> 1+ dup + cmove ( Stackelemente bewegen
           )
[VP2]  424:   drop drop
[V10]  425: ;
[000]  426:
[000]  427:
[CUF]  428: : d@ ( adr --> d ) ( doppeltgenauen Wert au
           s Speicher holen )
[545]  429:   dup @       ( Low Word )
[7P5]  430:   swap 2+ @   ( High Word )
[V10]  431: ;
[000]  432:
[000]  433:
[GGH]  434: : d! ( d adr --> ) ( doppeltgenauen Wert in
           Speicher schreiben )
[656]  435:   rot over !   ( Low Word )
[VS4]  436:   2+ !         ( High Word )
[V10]  437: ;
[000]  438:
[000]  439:
[000]  440:
[41E]  441: ( *** Unterstützung strukturierter Befehle
           *** )
[000]  442:
[6CD]  443: macro i ( Schleifenindex nach TOS kopieren
           )
[YMO]  444:   dup
[AD5]  445:   [ $89 $C8 ] ( mov ax,cx )
[V10]  446: ;
[000]  447:
[000]  448:
[TWA]  449: macro j ( zweiter Schleifenindex )
[K3O]  450:   r
[V10]  451: ;
[000]  452:
[000]  453:
[EK9]  454: macro leave ( Schleife verlassen )
[5C5]  455:   [ $89 $CF ] ( mov di,cx )
[V10]  456: ;
[000]  457:
[000]  458:
[L28]  459: macro exit ( Wort abbrechen )
[KE3]  460:   [ $C3 ] ( ret )
[V10]  461: ;
[000]  462:
[000]  463:
[000]  464:
[CD8]  465: ( *** Vergleichsoperatoren *** )
[000]  466:
[CT6]  467: : O= ( n --> flag ) ( Test, ob n=0 )
[8R4]  468:   [ $3D O O ] ( cmp  ax,O )
[MT4]  469:   [ $B8 O O ] ( mov  ax,O )
[KV3]  470:   [ $75 1 ]   ( jnz  +1   )
[6L4]  471:   [ $48 ]     ( dec  ax   )
[V10]  472: ;
[000]  473:
[000]  474:
[7T6]  475: : O< ( n --> flag ) ( Test, ob n<0 )
[8R4]  476:   [ $3D O O ] ( cmp  ax,O )
```

```
[MT4]  477:    [ $B8 0 0 ] ( mov  ax,0 )
[8X3]  478:    [ $7D 1 ]   ( jge  +1   )
[6L4]  479:    [ $48 ]     ( dec  ax   )
[V10]  480: ;
[000]  481:
[000]  482:
[2M9]  483: : 0<> 0= not ;  ( n --> flag ) ( Test, ob n
           <>0 )
[000]  484:
[000]  485:
[789]  486: : = - 0= ; ( n1 n2 --> flag ) ( Test, ob n1
           =n2 )
[000]  487:
[000]  488:
[089]  489: : < - 0< ; ( n1 n2 --> flag ) ( Test, ob n1
           <n2 )
[000]  490:
[000]  491:
[V6C]  492: : > swap - 0< ; ( n1 n2 --> flag ) ( Test,
           ob n1>n2 )
[000]  493:
[000]  494:
[L4A]  495: : <> = not ; ( n1 n2 --> flag ) ( Test, ob
           n1<>n2 )
[000]  496:
[000]  497:
[000]  498:
[005]  499: ( *** Funktionen *** )
[000]  500:
[818]  501: : abs ( n --> abs ) ( Absolutwert )
[ZZ5]  502:    dup 0< if minus endif
[V10]  503: ;
[000]  504:
[000]  505:
[M8B]  506: : sgn ( n --> signum ) ( Signum-Funktion )
[TA6]  507:    dup if dup abs / endif
[V10]  508: ;
[000]  509:
[000]  510:
[GF9]  511: : +- ( n1 n2 --> n ) ( berechnet sgn[n2]*n1
           )
[BD9]  512:    0< if abs minus else abs endif
[V10]  513: ;
[000]  514:
[000]  515:
[ASG]  516: : dminus ( d --> -d ) ( Vorzeichenwechsel,
           doppeltgenau )
[LX6]  517:    not swap not swap 1 0 d+
[V10]  518: ;
[000]  519:
[000]  520:
[79D]  521: : dabs ( d --> ud ) ( doppeltgenauer Absolu
           twert )
[5F6]  522:    dup 0< if dminus endif
[V10]  523: ;
[000]  524:
[000]  525:
[OZM]  526: : s->d ( n --> d ) ( einfachgenaue Zahl in
           doppeltgenaue konvertieren )
[XW5]  527:    dup 0< if -1 else 0 endif
[V10]  528: ;
[000]  529:
[000]  530:
[TN9]  531: : d+- ( d n --> n1 ) ( berechnet sgn[n]*d )
[LN9]  532:    0< if dabs dminus else dabs endif
[V10]  533: ;
[000]  534:
[000]  535:
[000]  536:
[000]  537:
[LN7]  538: ( *** einfache Ein-/Ausgabe *** )
[000]  539:
[000]  540:
[3LE]  541: macro (vector) ( für vektorisierte Ein-/Aus
           gabe )
[7R3]  542:    [ $E9 0 0 ]  ( jmp +0 )
[V10]  543: ;
[000]  544:
[000]  545:
[FXC]  546: : emit ( c --> ) ( TOS als ASCII-Zeichen au
           sgeben )
[BY2]  547:    (vector)
[A15]  548:    [ $88 $C2 ] ( mov  dl,al )
[NP4]  549:    [ $B4 $02 ] ( mov  ah,2  )
[5B4]  550:    [ $CD $21 ] ( int  21H   )
[TX2]  551:    drop out ++
[V10]  552: ;
[000]  553:
[000]  554:
[65F]  555: : inkey ( --> c ) ( Tastatureingabe --> TOS
           ; mit Echo )
```

```
[BY2]  556:    (vector)
[YMO]  557:    dup
[4M4]  558:    [ $B4 $01 ] ( mov  ah,1 )
[W84]  559:    [ $CD $21 ] ( int  21H  )
[WL4]  560:    [ $B4 $00 ] ( mov  ah,0 )
[V10]  561: ;
[000]  562:
[000]  563:
[CHA]  564: : key ( --> c ) ( Tastatureingabe --> TOS )
[BY2]  565:    (vector)
[YMO]  566:    dup
[WN4]  567:    [ $B4 $08 ] ( mov  ah,8 )
[W84]  568:    [ $CD $21 ] ( int  21H  )
[WL4]  569:    [ $B4 $00 ] ( mov  ah,0 )
[V10]  570: ;
[000]  571:
[000]  572:
[KV9]  573: : ?key ( --> flag ) ( Taste gedrückt ? )
[BY2]  574:    (vector)
[YMO]  575:    dup
[YP4]  576:    [ $B4 $0B ] ( mov  ah,11 )
[5B4]  577:    [ $CD $21 ] ( int  21H   )
[OT3]  578:    [ $98     ] ( cbw        )
[V10]  579: ;
[000]  580:
[000]  581:
[SW9]  582: : cr ( Zeilenvorschub ausführen )
[P45]  583:    13 emit 10 emit
[V10]  584: ;
[000]  585:
[000]  586:
[N2A]  587: : space ( Leerzeichen ausgeben )
[HR2]  588:    bl emit
[V10]  589: ;
[000]  590:
[000]  591:
[OFA]  592: : spaces ( n Leerzeichen ausgeben )
[D65]  593:    1 do space loop
[V10]  594: ;
[000]  595:
[000]  596:
[R37]  597: : count ( adr1 --> adr2 n )
[3X7]  598:    dup c@        ( Count-Byte holen )
[WF9]  599:    swap 1+ swap  ( Adresse erhöhen )
[V10]  600: ;
[000]  601:
[000]  602:
[KYB]  603: : type ( adr count --> ) ( String ausgeben
           )
[F9B]  604:    over + 1- swap  ( Endadresse berechnen )
[2AO]  605:    do
[4Z8]  606:      i c@ emit     ( Zeichen ausgeben )
[R21]  607:    loop
[V10]  608: ;
[000]  609:
[000]  610:
[CTE]  611: : " ( --> adr ) ( Adresse von String-Konsta
           nte nach TOS )
[DFC]  612:    r>              ( Adresse von Return-Stack
            holen )
[K49]  613:    dup dup c@      ( Count-Byte holen )
[NHB]  614:    + 1+            ( Rückkehr-Adresse berechn
           en... )
[APB]  615:    >r              ( ...und auf Return-Stack
           legen )
[V10]  616: ;
[000]  617:
[000]  618:
[V45]  619: : ." ( Text ausgeben )
[H8B]  620:    r>          ( Adresse von Return-Stack ho
           len )
[XKA]  621:    dup dup c@  ( Count-Byte holen
               )
[4VA]  622:    + 1+        ( Rückkehr-Adresse berechnen
           ... )
[ANA]  623:    >r          ( ... und wieder auf Stack le
           gen )
[T5A]  624:    count type  ( String ausgeben
               )
[V10]  625: ;
[000]  626:
[000]  627:
[L6F]  628: : (bs)  ( Backspace bei Eingabestring durch
           führen )
[C89]  629:    adr @ c@ dup      ( Count-Byte holen )
[N90]  630:    if
[FGB]  631:      1- adr @ c!     ( Count-Byte dekrementie
           ren )
[3WD]  632:      emit space bs   ( Backspace und Space au
           sgeben )
[A11]  633:    else
```

```
[6XD]  634:      drop drop 7     ( BEL auf Stack legen, f
       alls Count = 0 )
[FG1]  635:    endif
[V10]  636: ;
[000]  637:
[000]  638:
[KVC]  639: : (app)  ( Zeichen an Eingabestring anfügen
       )
[6L8]  640:    adr @ c@           ( Count-Byte holen )
[75C]  641:    dup ml c@ =        ( maximale Länge erreic
       ht ? )
[N90]  642:    if
[W1A]  643:      drop drop 7      ( BEL auf Stack legen )
[A11]  644:    else
[BS9]  645:      1+ adr @ c!      ( Count-Byte erhöhen )
[WBC]  646:      adr @ dup c@ +   ( Zeichenadresse ermitt
       eln )
[GC9]  647:      over swap c!     ( Zeichen ablegen )
[FG1]  648:    endif
[V10]  649: ;
[000]  650:
[000]  651:
[R2K]  652: : (get)  ( Zeichen von Tastatur für Eingabe
       string holen )
[586]  653:    key        ( Taste holen )
[M66]  654:    dup bs =   ( Backspace ? )
[N90]  655:    if
[9R0]  656:      (bs)
[A11]  657:    else
[XO7]  658:      dup 13 = ( Return-Taste ? )
[N90]  659:      if
[XZ8]  660:        drop 0     ( Null auf Stack legen )
[A11]  661:      else
[ZS9]  662:        dup 32 <   ( anderes Steuerzeichen ? )
[N90]  663:        if
[086]  664:          drop 7   ( BEL auf Stack )
[A11]  665:        else
[761]  666:          (app)
[FG1]  667:        endif
[FG1]  668:      endif
[FG1]  669:    endif
[V10]  670: ;
[000]  671:
[000]  672:
[ZAC]  673: : expect ( adr count --> ) ( String einlese
       n )
[BY2]  674:    (vector)
[91A]  675:    ml !               ( maximale Eingabelänge
       )
[GYA]  676:    0 over c!          ( Count auf Null setzen
       )
[7K9]  677:    adr !              ( Adresse in adr merken
       )
[BH1]  678:    begin
[MYB]  679:      (get) dup emit ( Zeichen holen und Echo
       )
[WU8]  680:      0=               ( bis CR gedrückt wird
       )
[DL1]  681:    until
[TO9]  682:    adr @ dup c@ +    ( letztes Zeichen )
[YN9]  683:    1+ 0 swap c!      ( Nullbyte anhängen )
[V10]  684: ;
[000]  685:
[000]  686:
[A5A]  687: : query ( Zeile in TIB einlesen )
[U06]  688:    tib @         ( Adresse holen )
[NZ8]  689:    80 expect     ( bis 80 Zeichen lesen )
[HB8]  690:    0 in c!       ( IN auf Null setzen )
[V10]  691: ;
[000]  692:
[000]  693:
[000]  694:
[U58]  695: ( *** formatierte Ausgabe *** )
[000]  696:
[9E9]  697: : <# ( Start formatierte Ausgabe )
[GSE]  698:    pad hld !       ( Zeiger hld auf Scratchpa
       d setzen )
[V10]  699: ;
[000]  700:
[000]  701:
[YGD]  702: : hold ( Zeichen an Adresse in hld speicher
       n )
[AW8]  703:    hld --     ( hld dekrementieren )
[MU7]  704:    hld @ c!   ( Zeichen ablegen    )
[V10]  705: ;
[000]  706:
[000]  707:
[UX6]  708: : # ( eine Ziffer ablegen )
[T59]  709:     ddup or              ( auf Null prüfen )
[N90]  710:    if
[2DA]  711:      base c@ m/mod       ( Ziffer berechnen )
```

```
[Y2A]  712:      rot '0' +          ( in ASCII-Format br
       ingen )
[ZGB]  713:      dup '9' >          ( Zahlenbasis größer
       10 ? )
[FW9]  714:      if 7 + endif       ( ja, dann Korrektur
       )
[A11]  715:    else
[ZX8]  716:      pad hld @ =        ( letzte Stelle ? )
[UL9]  717:      if                 ( ja, dann Null ausg
       eben )
[W7O]  718:        '0'
[A11]  719:      else
[Y2E]  720:        leading @        ( Leerzeichen oder N
       ull ausgeben )
[FG1]  721:      endif
[FG1]  722:    endif
[5M7]  723:     hold                ( Ziffer ablegen )
[V10]  724: ;
[000]  725:
[000]  726:
[YS9]  727: : #s ( restliche Ziffern ablegen )
[BH1]  728:    begin
[YAC]  729:      ddup or      ( Solange Zahl ungleich Nu
       l ... )
[BH1]  730:    while
[RG7]  731:      #            ( ... Ziffer ausgeben )
[X52]  732:    repeat
[V10]  733: ;
[000]  734:
[000]  735:
[18G]  736: : #> ( Ende formatierte Ausgabe, Zahl ist t
       ype-fertig )
[2H8]  737:    drop drop      ( Zahl entfernen )
[KA8]  738:    hld @ dup      ( hld-Adresse holen )
[WU8]  739:    pad - abs      ( Count-Byte erzeugen )
[V10]  740: ;
[000]  741:
[000]  742:
[7AG]  743: : sign ( Vorzeichen in formatierte Ausgabe
       einfügen )
[ZB9]  744:    rot      ( Vorzeichen-Byte holen )
[E18]  745:    0<       ( Vorzeichen negativ ? )
[N90]  746:    if
[SA8]  747:      '-'    ( ja, dann Minuszeichen )
[A11]  748:    else
[H68]  749:      bl     ( nein, dann Leerzeichen )
[FG1]  750:    endif
[HF7]  751:    hold     ( Zeichen ausgeben )
[V10]  752: ;
[000]  753:
[000]  754:
[FP9]  755: : d. ( doppelt genaue Zahl ausgeben )
[FTB]  756:    swap over      ( Vorzeichen-Byte erzeugen )
[BK8]  757:    dabs           ( Absolutwert bilden )
[SG6]  758:    <# # #s        ( Zahl ausgeben )
[TS8]  759:    sign #>        ( Vorzeichen anfügen )
[MZ6]  760:    type           ( ... und drucken )
[V10]  761: ;
[000]  762:
[000]  763:
[61H]  764: : d.r ( doppelt genaue Zahl in spez. Feldbr
       eite ausgeben )
[BP7]  765:    >r             ( Feldbreite merken )
[FTB]  766:    swap over      ( Vorzeichen-Byte erzeugen )
[BK8]  767:    dabs           ( Absolutwert bilden )
[9R7]  768:    r> 1-          ( Feldbreite holen )
[4P5]  769:    <# 1           ( Start Ausgabe )
[F4D]  770:    do # loop      ( Ausgabe entsprechend Feldb
       reite )
[TS8]  771:    sign #>        ( Vorzeichen anfügen )
[MZ6]  772:    type           ( ... und drucken )
[V10]  773: ;
[000]  774:
[000]  775:
[BH9]  776: : . ( einfach genaue Zahl ausgeben )
[ZLD]  777:    s->d    ( in doppelt genaue Zahl konvertie
       ren )
[326]  778:    d.      ( ... und ausgeben )
[V10]  779: ;
[000]  780:
[000]  781:
[HVF]  782: : .r ( einfach genaue Zahl in spez. Feldbre
       ite ausgeben )
[N5F]  783:    swap s->d    ( in doppelt genaue Zahl konv
       ertieren )
[1X7]  784:    rot d.r      ( ... und ausgeben )
[V10]  785: ;
[000]  786:
[000]  787:
[E6A]  788: : u. ( vorzeichenlose Zahl ausgeben )
```

```
[OTD]  789:    0           ( auf doppelte Genauigkeit erw
          eitern )
[GCC]  790:    <# # #s #>  ( Konvertierung ohne Vorzeiche
          n )
[D67]  791:    type        ( ... und ausgeben )
[V10]  792: ;
[000]  793:
[000]  794:
[82H]  795: : u.r ( vorzeichenlose Zahl in spez. Feldbr
          eite ausgeben )
[2KE]  796:    0 swap      ( auf doppelte Genauigkeit e
          rweitern )
[4P5]  797:    <# 1        ( Start Ausgabe )
[F4D]  798:    do # loop   ( Ausgabe entsprechend Feldb
          reite )
[TZ7]  799:    #> type     ( ... und drucken )
[V10]  800: ;
[000]  801:
[000]  802:
[000]  803:
[K99]  804: ( *** verschiedene Funktionen *** )
[000]  805:
[9LC]  806: : ? ( Inhalt eines Speicherworts ausgeben )
[A80]  807:    @ .
[V10]  808: ;
[000]  809:
[000]  810:
[3V5]  811: : char ( adr1 --> adr2 c )
[6ZD]  812:    begin ( Zeichen holen, Leerzeichen überle
          sen )
[4P6]  813:      dup c@     ( Zeichen holen )
[SZ6]  814:      dup bl =   ( Leerzeichen ? )
[BH1]  815:      while
[D68]  816:        drop     ( Zeichen wegwerfen )
[AS6]  817:        1+       ( Adresse erhöhen )
[X52]  818:    repeat
[V10]  819: ;
[000]  820:
[000]  821:
[G6G]  822: : digit ( c --> ? flag )  ( Zeichen in Zahl
          konvertieren )
[6HA]  823:    '0' -           ( Zeichen mit '0' vergleic
          hen )
[AN5]  824:    dup 0<          ( kleiner ? )
[N90]  825:    if
[D09]  826:      drop false    ( ja, keine Ziffer )
[ZY4]  827:      exit          ( Abbruch )
[FG1]  828:    endif
[LOA]  829:    dup 9 >         ( Zeichen größer '9' ? )
[N90]  830:    if
[YE8]  831:      7 - dup       ( ja, Hex-Korrektur )
[7W5]  832:      10 <          ( kleiner als 10 ? )
[N90]  833:      if
[1U8]  834:        drop false  ( ja, keine Ziffer )
[FM4]  835:        exit        ( Abbruch )
[FG1]  836:      endif
[FG1]  837:    endif
[G1B]  838:    dup base c@ <   ( mit base vergleichen )
[N90]  839:    if
[A58]  840:      true          ( ok, wenn kleiner )
[A11]  841:    else
[MXB]  842:      drop false    ( sonst keine gültige Ziff
          er )
[FG1]  843:    endif
[V10]  844: ;
[000]  845:
[000]  846:
[SUH]  847: : (number) ( d1 adr1 --> d2 adr2 ) ( Zeiche
          nfolge in Zahl wandeln )
[BH1]  848:    begin
[VEA]  849:      in ++             ( Eingabe-Zähler erhöh
          en )
[358]  850:      dup >r            ( Adresse merken )
[5YB]  851:      c@ digit          ( Zeichen holen und pr
          üfen )
[258]  852:      while             ( solange Ziffer )
[421]  853:        swap
[6BD]  854:        base @ u*       ( High-Word mit Base m
          ultiplizieren )
[AN2]  855:        drop rot
[LED]  856:        base @ u*       ( Low-Word mit Basis m
          ultiplizieren )
[UB8]  857:        d+              ( Ziffer aufaddieren )
[VY7]  858:        r> 1+           ( nächste Adresse )
[X52]  859:    repeat
[E70]  860:    r>
[V10]  861: ;
[000]  862:
[000]  863:
[6AM]  864: : number ( adr --> d ) ( String in doppeltg
          enaue Zahl konvertieren )
```

```
[BMB]  865:    1+ char         ( Leerzeichen überlesen )
[DE7]  866:    '-' =           ( negative Zahl ? )
[MM5]  867:    dup >r          ( merken )
[N90]  868:    if
[77C]  869:      1+ char drop  ( Minuszeichen überlesen
          )
[FG1]  870:    endif
[059]  871:    0 dup ddup      ( 4 Nullen auf Stack )
[AT9]  872:    dpl !           ( Dezimalpunkt löschen )
[RHB]  873:    in ! rot        ( Eingabezähler löschen )
[TX8]  874:    (number)        ( konvertieren )
[N38]  875:    dup c@          ( Zeichen holen )
[877]  876:    '.' =           ( Dezimalpunkt ? )
[N90]  877:    if
[6Z8]  878:      in @ dpl !    ( Position merken )
[8T7]  879:      1+            ( nächste Adresse )
[B7A]  880:      (number)      ( weiter konvertieren )
[FG1]  881:    endif
[LU8]  882:    drop            ( Adresse entfernen )
[5S7]  883:    r>              ( Vorzeichen holen )
[4WD]  884:    if dminus endif ( evtl. Vorzeichenwechsel
          )
[V10]  885: ;
[000]  886:
[000]  887:
[0A8]  888: : hex ( Zahlenbasis 16 setzen )
[0G2]  889:    16 base !
[V10]  890: ;
[000]  891:
[000]  892:
[GPA]  893: : decimal ( Zahlenbasis 10 setzen )
[MF2]  894:    10 base !
[V10]  895: ;
[000]  896:
[000]  897:
[RT6]  898: : .s ( Stack ausdrucken )
[7A0]  899:    cr
[7R8]  900:    base @          ( Zahlenbasis sichern )
[RE8]  901:    decimal         ( Ausgabe dezimal )
[706]  902:    sp@             ( Adresse holen )
[1V1]  903:    2+ dup
[TD5]  904:    s0 @ 2- =       ( Stack leer ? )
[N90]  905:    if
[PM3]  906:      ." STACK EMPTY"
[A11]  907:    else
[HU1]  908:      ." TOS>"
[BH1]  909:      begin
[V86]  910:        dup ? space  ( ausgeben )
[1W7]  911:        2+ dup       ( nächste Adresse )
[5E4]  912:        s0 @ 4 - >   ( fertig ? )
[DL1]  913:      until
[142]  914:      ." <BOTTOM"
[FG1]  915:    endif
[5L2]  916:    drop cr
[SK7]  917:    base !  ( Base restaurieren )
[V10]  918: ;
[000]  919:
[000]  920:
[K6B]  921: : min ( n1 n2 --> min )  ( Minimum von n1,
          n2 )
[ZK1]  922:    ddup >
[T84]  923:    if swap endif
[N21]  924:    drop
[V10]  925: ;
[000]  926:
[000]  927:
[A8B]  928: : max ( n1 n2 --> max )  ( Maximum von n1,
          n2 )
[LK1]  929:    ddup <
[T84]  930:    if swap endif
[N21]  931:    drop
[V10]  932: ;
[000]  933:
[000]  934:
[000]  935:
[4LB]  936: (save.voc)  ( Vocabulary Forth sichern )
[000]  937:
[000]  938:
[000]  939:
[PS2]  940: ( *** main *** )
[000]  941:
[6Z1]  942: : main
[NL6]  943:    7 emit ." Hello World"
[V10]  944: ;
```

Listing 2. Das Standard-Vocabulary von SForth

Strings mit SForth

Um mit einer Programmiersprache komfortabel zu arbeiten, sind Funktionen zur Manipulation von Zeichenketten unbedingt erforderlich. Mit dem hier vorgestellten Stringvokabular erweitert sich das Anwendungsspektrum des SForth-Compilers ganz erheblich.

Das Vocabulary »string« (Listing 1) lehnt sich in der Art und Weise der Stringbehandlung so weit wie möglich an entsprechende Spracherweiterungen von Standard-Forth an. Andererseits sind aus Gründen der übersichtlichen Anwendung dieses Vokabulars eine Reihe von Worten enthalten, die sich in Bezeichnung und Wirkung an Standard-Basic-Funktionen anlehnen. Die Tabelle zeigt alle neuen Worte des String-Vokabulars.

Grundsätzlich werden Strings bei SForth wie im Standard-Forth-Format behandelt. Ein String besteht daher aus einem führenden Byte, das die aktuelle Stringlänge angibt, gefolgt von den einzelnen Zeichen des Stringtexts. Demzufolge dürfen Strings maximal 255 Zeichen enthalten. Sie werden bearbeitet, indem die Adresse des Längenbytes auf den Stack gelegt wird. Diese Adresse dient den einzelnen Worten zur Stringbearbeitung als Basis. Wenn im String-Vocabulary also ein String angesprochen wird, dann ist immer gemeint, daß die String-Adresse auf dem Stack liegt.

Um komfortabel mit Strings arbeiten zu können, sind Stringvariable eine absolute Notwendigkeit. Stringvariable ohne Änderungen am Compilerkern implementieren zu können, erfordert jedoch ein etwas trickreiches Vorgehen.

Stringvariable werden in SForth grundsätzlich auf dem Heap, also in dem reservierten Speicherbereich, zwischen dem Ende des Dictionaries und dem unteren Ende des Datenstacks abgelegt. Der auf dem Heap zur Verfügung stehende freie Speicherplatz beträgt in der SForth-Version 3.1 normalerweise nur einige Dutzend Byte. Für das Anlegen von Stringvariablen muß also die Heapgröße durch die M-Option beim Aufruf des Compilers vergrößert werden.

Zum Anlegen von Stringvariablen dient der Befehl »string«. Wie bereits gesagt, arbeitet dieser Befehl etwas trickreich, so daß dazu einige zusätz-

MICROSOFT WORKS IN DEUTSCH.

Junge, Junge, als PC hat man das vielleicht schwer. Da wirst du mit Programmen gefüttert, und keiner fragt dich, ob du's auch verdaust.

Aber wie willst du dich verständlich machen, dem User sagen, was er wirklich braucht? Doch eines Tages, wie ein Blitz aus heiterem Himmel, hat's mein User geschnallt. Kann sein, daß ihm einer ein Zeichen gegeben hat – auf jeden Fall hab' ich jetzt mein Microsoft Works.

BESUCHEN SIE UNS AUF DER **CEBIT '88** VOM 16.3.–23.3.

Integriertes Paket: Tabellenkalkulation mit Grafikfunktionen – Textverarbeitung mit Serienbrieffunktion und Funktion zum Ausdruck von Etiketten – Datenbank mit Berichtsfunktionen – Kommunikationsmodul – Integriertes Lernprogramm – Komfortable Datenaustauschmöglichkeiten.

Eine komplette Grundausstattung, die alle unsere Bedürfnisse erfüllt. Textverarbeitung, Tabellenkalkulation, Grafik, Datenbank und Kommunikation – und natürlich in Deutsch. Alles dabei, was man haben muß, um täglich klar zu kommen: Datenaustausch zwischen den Modulen, Grafiken in den Text mischen, Adressen aus der Datenbank im Serienbrief und, und, und ...

Natürlich alles in einem Programm – so miteinander zu kombinieren, wie's sein soll. Ganz klar, daß auch ein Lernprogramm dabei ist – das erspart mir eine Menge wilde Fummelei. Aber das ist eigentlich gar nicht nötig, denn Microsoft Works macht's meinem User sowieso besonders leicht.

Also, PCs – zeigt doch Euren Usern auch was Sache ist. Dann bekommt Ihr, was Ihr braucht.

MS/DOS CBT 3½ 5¼

Microsoft®

ZUKUNFT DER SOFTWARE

COUPON

Bitte senden Sie mir Informationsmaterial zu Microsoft Works. Ich nutze Software: ☐ privat ☐ beruflich/Branche _____

Mein Rechner: ☐ MS-DOS ☐ MS-OS/2 ☐ Macintosh

Bitte senden Sie den Coupon an: Microsoft GmbH · Erdinger Landstraße 2 · 8011 Aschheim-Dornach
Absender nicht vergessen. DOSE 1/88

liche Erläuterungen sinnvoll sind. Die Definition des Wortes »string« finden Sie ganz zu Anfang im Listing des String-Vocabularies. Der Aufruf lautet:

```
<Variablename> <Länge> string
```

Es wird also zunächst der Name einer ganz normalen Variablen benutzt. Diese Variable müssen Sie natürlich zuvor also solche deklarieren. Durch den Variablennamen wird beim Compilieren Code erzeugt, um die Adresse der Variablen auf den Stack zu legen. Dieser vom Compiler erzeugte Code sieht so aus:

```
dec si
dec si
mov [si],ax
mov ax,$+3
ret
defw <Variable>
```

Der Code bringt das AX-Register, das den TOS enthält, auf den durch SI simulierten Datenstack und lädt es anschließend mit der Adresse der Variablen. Der Trick besteht nun darin, daß im Wort »string« die Adresse der einfachen Variablen um drei vermindert wird. Somit steht ein Zeiger auf der Programmstelle drei Byte vor der Variablen zur Verfügung. Diese Programmstelle ist aber gerade die im Befehl »mov ax,$ + 3« enthaltene Adressenangabe. Diese Adressenangabe wird dann einfach mit der Heap-Adresse der Stringvariablen überschrieben. Jeder Aufruf der Variablen liefert anschließend nicht mehr die Adresse des eigentlichen Variablen-Datenwortes, sondern die Heap-Adresse der assoziierten Stringvariablen.

Die Wirkung des Worts »string« hier im einzelnen: Der Heap-Speicherplatz wird in SForth über die Variable DP (Dictionary-Pointer) verwaltet. Das Wort »here« liefert den aktuellen Wert des Dictionary-Pointers. Mit der Befehlsfolge »here swap« wird der aktuelle Stand des Dictionary-Pointers geholt und mit der zuvor auf dem Stack liegenden Stringlänge vertauscht. Bei der Längenangabe handelt es sich um die maximale Länge des zu definierenden Strings. Die tatsächliche Länge wird zu Null initialisiert. Dies erledigt der Befehl »0 c,«, durch den ein Byte mit dem Wert Null auf dem Heap reserviert wird. »allot« reserviert im nächsten Schritt die angegebene maximale Stringlänge. Der Dictionary-Pointer zeigt anschließend auf das nächste freie Byte hinter dem soeben definierten String. Der ursprüngliche Wert des Dictionary-Pointers wurde anfangs ja auf dem Stack festgehalten. Er repräsentiert nun die Angfangsadresse, unter der der neue String zugänglich ist. Durch die abschließende Manipulation mit »swap 3 - !« wird diese Stringadresse

als neue Variablenadresse eingesetzt. Wenn Sie also beispielsweise im Hauptprogramm eine Stringvariable namens »alpha« mit einer maximalen Länge von 120 Zeichen definieren wollen, dann lautet der entsprechende Aufruf unter SForth:

```
variable alpha 0
: main alpha 120 string ...
```

Listing 2 zeigt als Demo-Programm einige Anwendungen des String-Vokabulars.

Besonders interessant sind die Worte »nfa« und »cfa«. »nfa« erwartet eine Stringadresse auf dem Stack und durchsucht das gesamte SForth-Dictionary nach einem gleichlautenden SForth-Befehl.

Zurückgeliefert wird die Namensfeldadresse des entsprechenden Worts, wenn ein solches Wort existiert, andernfalls eine Null. Durch das Wort »cfa« wird aus einer solchen Namensfeldadresse die zugehörige Codefeldadresse berechnet.

»nfa« liefert die Dictionary-Adresse, ab der der Name des entsprechenden Worts gespeichert ist, »cfa« berechnet daraus die Adresse des ausführbaren Codes des Worts.

(ev)

string	(str n – >)	Stringvariable definieren
len	(str – > n)	Stringlänge
$.	(str – >)	String drucken
$!	(str1 str2 – >)	str1 an Position str2 speichern
$+	(str1 str2 – >)	str1 an Ende von str1 anhängen
left$	(str n – >)	n linke Zeichen von str
mid$	(str pos n – >)	n Zeichen von str ab pos
right$	(str n – >)	die n letzten Zeichen von str
clr$	(str – >)	Leerstring erzeugen
val	(str – > n)	String in Zahl wandeln
str$	(n – > str)	Zahl in String wandeln
$=	(str1 str2 – > flag)	Strings vergleichen
$pos	(str c – > n)	Zeichen c in str suchen
nfa	(str – > adr)	Namensfeldadresse aus Namen
cfa	(adr1 – > adr2)	Codefeldadresse aus nfa bilden

Tabelle der neuen Worte im String-Vocabulary

```
[1CE]    0: ( String-Vocabulary für SForth, Version 3.1
            )
[000]    1:
[MF4]    2: include forth
[000]    3:
[3V5]    4: vocabulary string
[000]    5:
[000]    6:
[CBF]    7: : string ( str n --> )  ( Stringvariable de
            finieren )
[84B]    8:    here swap  ( Adresse holen und merken )
[1TA]    9:    0 c,       ( Count-Byte initialisieren )
[BXB]   10:    allot      ( Platz für String reservieren
            )
[G5A]   11:    swap 3 - ! ( Variablenaufruf patchen )
[V10]   12: ;
[000]   13:
[000]   14:
[278]   15: : len ( str --> n )  ( Stringlänge )
[370]   16:    c@
[V10]   17: ;
[000]   18:
[000]   19:
[UW7]   20: : $. ( str --> )  ( String drucken )
[2Y2]   21:    count type
[V10]   22: ;
[000]   23:
[000]   24:
[9F4]   25: : $! ( str1 str2 --> )
[PPF]   26:    over len 1+ cmove  ( str1 an Position str
            2 speichern )
[V10]   27: ;
[000]   28:
[000]   29:
[H9E]   30: : $+ ( str1 str2 --> )  ( str1 an Ende von
            str1 anhängen )
[978]   31:    swap count >r ( Länge retten )
[177]   32:    over count +  ( Stringende )
[FN8]   33:    r cmove       ( Bytes kopieren )
[GV6]   34:    dup len r> +  ( neue Länge )
[056]   35:    swap c!       ( eintragen )
[V10]   36: ;
[000]   37:
[000]   38:
[3TD]   39: : left$ ( str n --> )  ( n linke Zeichen vo
            n str )
[687]   40:    over len min  ( neue Länge )
[056]   41:    swap c!       ( eintragen )
[V10]   42: ;
[000]   43:
[000]   44:
[ZND]   45: : mid$ ( str pos n --> )  ( n Zeichen von s
            tr ab pos )
[HZ7]   46:    >r over len   ( Stringlänge )
[OE5]   47:    ddup >        ( pos > len ? )
[N90]   48:    if
[V66]   49:      drop drop   ( pos, len )
[XL6]   50:      0 swap c!   ( Leerstring )
[A11]   51:    else
[9L5]   52:      over -      ( len - pos )
[TE6]   53:      r> min >r   ( neue Länge )
[YB4]   54:      over +      ( Quelle )
[4P3]   55:      over 1+     ( Ziel )
[WM6]   56:      r cmove     ( kopieren )
[CO7]   57:      r> swap c!  ( neue Länge )
[FG1]   58:    endif
[V10]   59: ;
[000]   60:
[000]   61:
[EXG]   62: : right$ ( str n --> )  ( die n letzten Zei
            chen von str )
[YG9]   63:    over len min  ( Länge festlegen )
[DM8]   64:    >r dup len    ( aktuelle Länge )
[8T5]   65:    r - over + 1+ ( Quelle )
[Y44]   66:    over 1+       ( Ziel )
[GS6]   67:    r cmove       ( kopieren )
[4T6]   68:    r> swap c!    ( neue Länge )
```

```
[V10]   69: ;
[000]   70:
[000]   71:
[32B]   72: : clr$ ( str --> )  ( Leerstring erzeugen )
[B92]   73:    0 swap c!
[V10]   74: ;
[000]   75:
[000]   76:
[UFB]   77: : val ( str --> n )  ( String in Zahl wande
            ln )
[C23]   78:    number drop
[V10]   79: ;
[000]   80:
[000]   81:
[WRB]   82: : str$ ( n --> str )  ( Zahl in String wand
            eln )
[CRB]   83:    s->d          ( doppeltgenaue Zahl erzeugen
            )
[4U8]   84:    swap over  ( Vorzeichen-Byte )
[3A6]   85:    dabs       ( Absolutwert )
[TN9]   86:    <# # #s    ( Zahlenstring erzeugen )
[ZF6]   87:    sign #>    ( Vorzeichen )
[T39]   88:    over 1- c! ( Count-Byte ablegen )
[B40]   89:    1-
[V10]   90: ;
[000]   91:
[000]   92:
[5SC]   93: : $= ( str1 str2 --> flag ) ( Strings vergl
            eichen )
[X4B]   94:    over c@ over c@ min  ( Anzahl Vergleiche
            )
[E01]   95:    dup 0
[2A0]   96:    do
[N21]   97:      drop
[P09]   98:      over c@ over c@     ( Bytes holen )
[EU6]   99:      = not               ( ungleich ? )
[N90]  100:      if
[RVA]  101:        false leave       ( Schleife verlassen
            )
[A11]  102:      else
[W1B]  103:        1+ swap 1+ swap   ( Adressen erhöhen )
[TA6]  104:        true              ( True-Flag )
[FG1]  105:      endif
[R21]  106:    loop
[VNC]  107:    rot drop swap drop    ( Adressen vernichte
            n )
[V10]  108: ;
[000]  109:
[000]  110:
[FYC]  111: : $pos ( str c --> n )  ( Zeichen c in str
            suchen )
[W8A]  112:    over len dup 1 ( Schleifengrenzen )
[2A0]  113:    do
[N21]  114:      drop
[5V8]  115:      over i +      ( Zeichen adressieren )
[T49]  116:      c@ over =     ( Zeichen gefunden ? )
[N90]  117:      if
[LW7]  118:        i leave     ( Zähler merken )
[A11]  119:      else
[G10]  120:        0
[FG1]  121:      endif
[R21]  122:    loop
[ZD5]  123:    rot drop swap drop
[V10]  124: ;
[000]  125:
[000]  126:
[05E]  127: : nfa ( str --> adr )  ( Namensfeldadresse
            aus Namen )
[KTB]  128:    dictionary        ( Beginn Dictionary-Link
            )
[BH1]  129:    begin
[M97]  130:      dup 2+          ( zeigt auf Name )
[T4E]  131:      3 pick $= not   ( String mit Namen vergl
            eichen )
[BH1]  132:    while
[B67]  133:      @               ( neuen Zeiger holen )
[HN7]  134:      dup @ 0=        ( Dictionary-Ende ? )
```

```
[N90]  135:        if
[U1A]  136:          drop false ( ja, Rückkehr mit Null
         )
[031]  137:          exit
[FG1]  138:        endif
[X52]  139:    repeat
[2AA]  140:    2+                ( Namensfeldadresse bild
         en )
[V10]  141: ;
[000]  142:
[000]  143:
[B6F]  144: : cfa ( adr1 --> adr2 )  ( Codefeldadresse
         aus nfa bilden )
[G82]  145:    count + 1+
[V10]  146: ;
[000]  147:
[000]  148:
[5S2]  149: (save.voc)
[000]  150:
[000]  151:
[6Z1]  152: : main
[V10]  153: ;
```

Listing 1. Das Vocabulary »string« zu SForth 3.1

```
[HFD]    0: ( Demo zum String-Vocabulary von SForth 3.1
           )
[000]    1:
[MF4]    2: include forth
[735]    3: include string
[000]    4:
[NG3]    5: variable a$ 0
[RG3]    6: variable b$ 0
[000]    7:
```

```
[K12]    8: : init
[8R2]    9:    a$ 80 string
[9R2]   10:    b$ 80 string
[7B9]   11:    ." ** String-Demo SForth 3.1 **" cr cr
[L6D]   12:    ." Bitte geben Sie einen Text ein: " quer
           y cr
[V10]   13: ;
[000]   14:
[6T2]   15: : ausgabe
[612]   16:    a$ len 1
[2AO]   17:    do
[OFA]   18:      a$ i + c@  ( Zeichen aus String holen )
[68F]   19:      emit bl emit  ( Zeichen und Leerzeichen
           ausgeben )
[7L2]   20:    loop cr
[V10]   21: ;
[000]   22:
[FT2]   23: : links
[712]   24:    b$ len 1
[2AO]   25:    do
[C17]   26:      b$ a$ $!    ( a$ restaurieren )
[2BA]   27:      a$ i left$ ( Zeichenkette erzeugen )
[NN6]   28:      a$ $. cr    ( ...und ausgeben )
[R21]   29:    loop
[V10]   30: ;
[000]   31:
[FR2]   32: : rechts
[712]   33:    b$ len 1
[2AO]   34:    do
[BK7]   35:      b$ a$ $!     ( a$ restaurieren )
[BAA]   36:      a$ i right$ ( Zeichenkette erzeugen )
[TZ6]   37:      a$ $. cr     ( ...und ausgeben )
[R21]   38:    loop
[V10]   39: ;
[000]   40:
[6Z1]   41: : main
[G21]   42:    init
[1M8]   43:    tib @ a$ $!  ( Text in a$ speichern )
[245]   44:    a$ b$ $!     ( ...und in b$ )
[CN3]   45:    links rechts
[ECB]   46:    cr ." Sie haben eingegeben: " cr ausgabe
[215]   47:    ." ... und das war's !" cr
[V10]   48: ;
```

Listing 2. Ein kleines Demo-Programm zum String-Vocabulary

PC – Kasse

E/A Überschußrechnung für PC/XT/AT kompatible mit 512 KByte, auch für SCHNEIDER PC. Schluß mit mühevollem Suchen, Sortieren und Rechnen... PC–Kasse übernimmt die Arbeit!

Wünschen Sie sich ein Kassenbuch, das
* praktisch keine Buchführungskenntnisse voraussetzt?
* automatisch berechnet und prüft?
*in allen Feldern mit indiv. Filter zugleich sucht?
* nach dem Datum sortiert, und Belegnr. ersetzt?
* Reports nach mehreren verschiedenen Kriterien erstellt?
* superschnell, flexibel, einfach und preiswert ist?

Belege eingeben, ändern, kopieren, suchen, löschen, blättern, auflisten, sortieren, herausfiltern, zählen, aufsummieren.

Tages-, Monats-, Quartals-, Konten-, Summen-, Gesamt-, Universal-Report auf Drucker oder Monitor mit Aufsummierung, Seitenzählung, Datum, nach verschiedensten Kriterien individuell sortiert oder/und gefiltert, u.s.w....

PC – KASSE kann's... und kostet nur DM 148, –

Info bei:

M.M.W. – Worms,
Joh.-Westermann-Platz 1,
4780 Lippstadt,
Händleranfragen willkommen!
Tel.: 0 29 41/5 92 90

Ihr Programm

PC Text2/3, Word Perfect, Word, Euroscript, Kon-Text, Xywrite III, Nota Bene 2.0, Textino Plus 2.0, PTS, PTS Publisher

korrigiert

bereits, während Sie Ihre Texte schreiben

und trennt

automatisch, justiert

richtig mit

CARLOS!

Weil CARLOS mehr als

450.000 Wörter

mit allen richtigen Trennungen in seinem Wörterbuch hat. 1 Million Wörter können Sie noch hinzufügen. Neugierig? Für

911 Mark

lernt Ihr PC Deutsch oder Englisch in 5 Minuten

Prospekt und Auto-Demo für IBM PC bei
ASSISTENT Datentechnik
Helmo Roth-Seefried
Postfach 11 11 · 5220 Waldbröl
Tel. 0 22 91 / 78 44

C-Tools

C-terp der prof. C-Interpreter
neu: für Turbo C nur.DM 350,–
BTREE + ISAM file management routines (Multi-User Option erhältlich).....**DM 350,–**
GraphiC wiss. Präsentationsgrafik jetzt neue **Version 4.1**.....**DM 855,–**
IP C-Druckerfunktionen für jeden Drucker.....**DM 225,–**
Vance C-lib Window Bibliothek (UNIX "curses" kompatibel)..**DM 295,–**
EGA-Tools im C und Assembler Source von COBRA.....**DM 195,–**
MID Treiber für beliebig viele V24-Schnittstellen.....**DM 285,–**
PC-lint C Syntax- und Semantikprüfung über mehrere Files.**DM 399,–**
VTEK Tektronix 4014 und VT 100
Terminal-Emulation...**DM 350,–**
PrintQ der Druckerspooler für den PC.....**DM 285,–**

DeSmet C Development Package
Komplettes C Entwicklungssystem mit sehr schnellem C-Compiler, Editor, Linker, Librarian **Source Code Debugger,** umfangreiche Standardbibliothek. Jetzt mit noch mehr Utilities und Grafik für CGA und Hercules mit Large Case Option DM 395,– Library Source DM 295,– **nur DM 295,–**

Viele Produkte mehr:
Info anfordern!
Keine zusätzlichen Versandkosten!

KESSLER Softwareentwicklung
Mitteldorfstr.17 3400 Göttingen Tel. 0551-792488

Die neuen Systemworte von SForth 3.1

Ab der Version 3.0 kennt der SForth-Compiler eine ganze Reihe neuer Systemworte insbesondere zur Interrupt-Bearbeitung, die den Zugriff auf die Betriebssystemebene erleichtern. Zusammen mit einem Spezial-Vocabulary ist damit aber auch die Programmierung speicherresidenter Utilities möglich.

Gegenüber den SForth-Versionen 2.x hat sich eine ganze Menge getan. Alle verzweigenden (strukturierenden) Systemworte wurden auf »Near Jumps« umgestellt, arbeiten jetzt also mit 16-Bit-Sprungadressen anstelle von 8-Bit-Sprüngen. Die Compiler-Funktion »short« für die Übersetzung von »Short Jumps« wurde daher durch eine neue Funktion »near« ersetzt. Die Verwendung von »Near Jumps« erlaubt mehrfach geschachtelte Strukturen über mehr als 127 Byte Distanz. Dadurch können sich Strukturen wie »if...else...endif« oder »do...loop« nun über praktisch beliebig lange Programmteile erstrecken. Eine Erhöhung des Maximalwertes für den Compiler-internen Stackpointer erlaubt dabei eine fast unbegrenzte Verschachtelung von Strukturworten.
Komplett neu ist auch die Einführung von Case. Diese Struktur erzeugt zwar kaum weniger Code als eine äquivalente Schachtelung von If-Abfragen, macht das Programm aber übersichtlicher. Listing 2 zeigt eine Zeichenkonvertierung, wie sie im Rahmen einer Druckeranpassung notwendig sein könnte; und zwar einmal in der Form einer mehrfachen If-Abfrage und ein zweites Mal unter Zuhilfenahme der neuen Case-Anweisung. Der Gewinn an Übersichtlichkeit ist sofort zu sehen.
Durch das Schlüsselwort »case« wird ein internes Case-Flag im Compiler gesetzt und der Of-Zähler mit Null initialisiert. Durch »of« erfolgt der Vergleich des aktuellen TOS (Top of Stack, also das oberste Stackelement) mit dem zweiten Stackelement (Second). Sind die beiden Werte gleich, werden die Befehle zwischen »of« und »endof« ausgeführt. »endof« ist völlig analog zu »else« aufgebaut und daher im Compiler intern als Synonym zu »else« definiert. Mit Rücksicht auf die Behandlung der Case-Anweisung in anderen Forth-Dialekten wurde zusätzlich das Wort »;;« als Synonym für »endof« eingeführt.
Nach dem Ende eines Of-Zweiges der Case-Anweisung wird die Ausführung des Programms in jedem Fall hinter »endcase« fortgesetzt. »endcase« schließt damit immer eine Case-Sequenz ab. Ähnlich des Vorgangs bei »endif« werden alle Vorwärtsreferenzen zu Sprungzielen aufgelöst.
Eine kleine Einschränkung hinsichtlich der Case-Anweisung ist jedoch vorhanden: In der jetzigen Version des Compilers sind keine verschachtelten Case-Anweisungen erlaubt.

Speicherresidente Utilities sind mit SForth 3.1 problemlos zu programmieren

Da der SForth-Compiler sehr schnellen und vor allem kompakten Code erzeugt, ist die Programmierung von Interrupt-Routinen ein besonders interessantes Anwendungsgebiet. Durch Anlage von Stacks und Datenbereiche belegt SForth in den bisher veröffentlichten Versionen 2.x immer ein ganzes 64 KByte-Segment. Für ein speicherresidentes Programm ist dieses Verhalten nicht erwünscht, zumal solche in SForth geschriebenen Utilities selten mehr als 10 KByte Code besitzen.
Durch die Modifikationen der Compiler-Version 3.1 werden nun die Stack- und Pufferbereiche direkt hinter dem compilierten Code angelegt. Der Benutzer muß mit der neuen Option »M« bei Aufruf des Compilers angeben, wieviel Speicherplatz für Allot-Kommandos auf dem Heap reserviert werden soll. Wird die M-Option nicht benutzt, dann besteht bei häufigem Einsatz von Allot-Befehlen (auch bei »,« und »c,«) die Gefahr einer Kollision zwischen dem Heap-Zeiger DP und dem Datenstack. Der Sicherheitsabstand zum Datenstack ist für normale Anwendungen großzügig bemessen – eine Schätzung des benötigten Speichers für die M-Option genügt daher.
Der TIB (Terminal Input Buffer) beginnt dann direkt hinter dem Programmende und ist 80 Zeichen groß. Es folgen die Bereiche für den PAD, reservierten Speicher (Heap) und Datenstack. Die korrespondierenden Systemvariablen werden vom Compiler gesetzt. Die im Forth-Vocabulary gesetzten Initialwerte werden dabei überschrieben. Beim Programmstart wird der Datenstackpointer (das SI-Register) mit dem Inhalt der Variablen »s0« initialisiert. Die Lage des Return-Stacks verändert sich dabei nicht, da sie bei normal ablaufenden Programmen gleichgültig ist. Bei speicherresident gemachten Interrupt-Programmen wird der Stack der aufrufenden Ebene mitbenutzt.

Die neuen Systemworte für die Interrupt-Behandlung

Das Wort »int« bewirkt die Ausführung einer Int-Instruktion des Prozessors. Es wird ein Software-Interrupt ausgelöst. Die Prozessorregister AX, BX, CX und DX werden dabei mit den Informationen aus den korrespondierenden Variablen gesetzt. Die Interruptnummer wird auf dem TOS erwartet. Bei Rückkehr von der gerufenen Interrupt-Routine werden die Inhalte der Register AX bis DX wieder in die entsprechenden Variablen geschrieben. Dieses Systemwort erlaubt unkomplizierte Interrupt-Aufrufe in allen Fällen, wo sowohl die Aufruf- als auch die Rückgabeparameter nur in den Registern AX bis DX übergeben werden. In anderen Fällen können Spezialworte günstiger sein.
Das Wort »find« legt die Adresse des im SForth-Quelltext direkt folgenden Worts auf den TOS, wenn dieses bereits bekannt ist. Andernfalls bricht der Compiler mit einer Fehlermeldung ab. Die ermittelte Adresse ist die Wortfeldadresse, zeigt also auf das Längenbyte des Wortnamens.
Das Systemwort »beginint« ist wichtig für die Programmierung speicherresidenter Utilities. Es muß am Anfang einer Interrupt-Service-Routine stehen. Das Wort überträgt die Registerinhalte von AX bis DX in die entsprechenden Variablen. Gleichzeitig werden alle Register auf den Stack gerettet. Gegebenenfalls ist es erforderlich, die Inhalte der Registervariablen während der Ausführung der Interrupt-Routine nochmals zwischenzuspeichern; jedoch nur, wenn die Variableninhalte innerhalb der Routine verändert werden, am Ende aber die alten Werte zurückgegeben werden müssen. Das ist beispielsweise dann der Fall,

ax	(– > adr)	Register AX
bx	(– > adr)	Register BX
cx	(– > adr)	Register CX
dx	(– > adr)	Register DX
makevect	(var – >)	Long Pointer als Vektor einrichten
vecthi	(var – > adr)	Segment-Adresse eines Vektors holen
vectlo	(var – > adr)	Offset-Adresse eines Vektors holen
getvect	(n var – >)	Interruptvektor Nummer n holen und sichern
setvect	(n var – >)	Interruptvektor Nummer n setzen
ds@	(– > n)	DS-Registerinhalt auf TOS (Datensegment)
cs@	(– > n)	CS-Registerinhalt auf TOS (Codesegment)
keep	(– >)	Programm speicherresident machen

Tabelle der Worte im Isys-Vocabulary

wenn mehrere Interrupt-Service-Routinen gleichzeitig benutzt werden.

Das Wort »beginint« setzt ferner das DS-Register gleich dem CS-Register. Die Systemvariable »r0« wird mit dem Wert des SP-Registers geladen. Es wird also kein eigener Returnstack angelegt , sondern der aktuelle Rechnerstack einfach mitbenutzt.

Das Gegenstück zu »beginint« ist »endint«. Damit wird eine Interrupt-Routine ordnungsgemäß wieder verlassen. Alle zuvor geretteten Register werden wieder vom Stack geholt und die Register AX, BX, CX und DX aus den entsprechenden Registervariablen gesetzt. Zum Schluß wird die Interrupt-Routine mit einem Iret-Befehl beendet.

Zwei weitere Systemworte unterstützen die Interrupt-Programmierung. »exitto« bewirkt den Sprung in eine andere Interrupt-Routine ohne Rückkehr. Das Wort erwartet die Adresse einer Variablen auf dem TOS. Der Inhalt der Variablen muß auf einen Long Pointer (vier Byte) zeigen, der wiederum auf die anzuspringende Routine zeigen muß.

Das Wort »callto« ist ähnlich aufgebaut wie »exitto«, und auch sein Aufruf ist ähnlich. Im Gegensatz zu »exitto« erfolgt allerdings die Rückkehr in die aufrufende Routine. »callto« wird angewendet, wenn eine Interrupt-Routine nicht über eine Int-Instruktion zu erreichen ist, weil man zum Beispiel den zugehörigen Interrupt-Vektor »verbogen« hat. Nähere Einzelheiten zu dieser Thematik bringt der Kurs »Speicherresidente Utilities selbstgemacht« in der DOS International ab Ausgabe 2'88.

Die in SForth 3.1 implementierte Version von »callto« zerstört den Datenstackpointer. Bei der Rückkehr von einer mit »callto« aufgerufenen Routine wird der Datenstack daher neu initialisiert. Alle Werte, die vor dem Aufruf noch auf dem Stack waren, sind verloren, ebenso die vor dem Aufruf gültigen Registerinhalte. Bei Bedarf müssen diese Werte zwischengespeichert und anschließend restauriert werden.

Wegen der Art der Stackbehandlung dürfen Sie die Worte »endint«, »exitto« und »callto« nur innerhalb einer mit »beginint« eröffneten Interrupt-Routine verwenden. Alle vier Systemworte unterstützen die schnelle Entwicklung von Interrupt-Routinen, wie sie bei speicherresidente Utilities zum Einsatz kommen. Allerdings ist die Implementation dieser Worte recht umfangreich. Wenn der Programm-Overhead durch das Sichern aller Register und die Verwendung von Defaultvariablen nicht mehr tragbar erscheint, können Sie ohne weiteres auf diese Hilfen verzichten. Die Realisierung als Systemworte beansprucht ja keinen Platz im Dictionary.

Das Vocabulary »isys« (Listing 1) ergänzt die grundlegenden Systemworte zur Interrupt-Programmierung. Wie bereits bekannt, enthalten die Variablen AX, BX, CX und DX nach »beginint« die Inhalte der gleichnamigen Prozessorregister. Die Systemworte »int«, »exitto«, »callto« und »endint« verwenden die Inhalte dieser Registervariablen, um ihrerseits vor der Ausführung der eigentlichen Routine die Prozessorregister zu laden.

Das Wort »makevect« aus dem Isys-Vokabular erwartet die Adresse einer vorher deklarierten Variablen auf dem TOS. Es werden vier Byte im freien Speicherbereich (auf dem Heap) reserviert, um als Long Pointer zu dienen. Die Adresse dieses Long Pointers wird der Variablen zugewiesen, deren Adresse sich bei Aufruf von »makevect« auf dem Stack befand.

Das Spezialvokabular »isys« macht SForth zum idealen Entwicklungssystem für Interruptroutinen

»vecthi« und »vectlo« erwarten ebenfalls die Adresse einer Variablen auf dem Stack. Die Worte legen die High- beziehungsweise Low-Adresse des Long Pointers, auf den der Inhalt der Variablen zeigt, auf dem TOS ab.

»getvect« und »setvect« dienen zum Lesen beziehungsweise Setzen eines Interruptvektors. Beide Worte erwarten die Interruptnummer und die Adresse einer Variablen auf dem Stack. Der zur Interruptnummer gehörende Interruptvektor wird mit »getvect« in den Long Pointer, auf den die Variable zeigt, geschrieben. Mit »setvect« wird der Interruptvektor auf die Adresse, auf die dieser Long Pointer zeigt, gesetzt. Mit anderen Worten: Der Long Pointer wird zum neuen Interrupt-Vektor.

Ganz wesentlich ist das Wort »keep«, mit dem Sie das aktuelle Programm verlassen; es bleibt jedoch speicherresident (»Terminate and Stay Resident«). »keep« benötigt keine weiteren Parameter, sondern berechnet alle notwendigen Angaben wie zum Beispiel die Größe des residenten Programms völlig selbstständig. Dieses Wort ist nur als letzter Aufruf im Main-Wort sinnvoll.

Listing 2 zeigt ein kleines Beispiel für den Einsatz des Isys-Vokabulars. Mit diesem Listing können Sie dem Drucker ein »X« für ein »U« vormachen. Dazu wurde einfach eine kleine Routine (»printproc«) verwendet, die in den Drucker-Interrupt des BIOS eingehängt wird und die eine Umcodierung der an den Drucker zu sendenden Zeichen vornimmt. Diese Routine ist in der hier abgedruckten Form natürlich nur als kleines Beispiel zu sehen, läßt sich aber zu einem vollständigen Druckertreiber mit Zeichen-Umcodierung erweitern. Das Main-Wort führt die Installation des speicherresidenten Teils durch. Eine vollständige Erklärung aller notwendigen Schritte führte hier etwas zu weit, daher sei an dieser Stelle nochmals auf den Kurs »Speicherresidente Programme selbstgemacht« in der DOS International hingewiesen.

(Erwin Schomburg/ev)

```
[30A]    0: ( Vokabular ISYS für Interrupt-    )
[OPA]    1: ( handling und TSR Programmierung )
[15B]    2: ( nur mit modifiziertem Compiler  )
[E68]    3: ( ab Version 3.0 verwenden !      )
[BD7]    4: ( [c] Erwin Schomburg             )
[000]    5:
[000]    6:
[9HP]    7: include forth  ( Achtung, nur Forth-Vocabul
           ary ab Version 1.4 verwenden ! )
[000]    8:
[ZT5]    9: vocabulary isys
[000]   10:
[29F]   11: variable ax 0        ( Diese Variablen korr
```

```
           espondieren  )
[H5F]   12: variable bx 0        ( mit den gleichnamige
           n Prozessor- )
[ASA]   13: variable cx 0        ( Registern !
                       )
[RS3]   14: variable dx 0
[000]   15:
[X3G]   16: : makevect ( var --- )              ( 4 Byt
           e im Heap reservieren   )
[97E]   17:                here swap ! 4 allot   ( Varia
           ble zeigt auf diesen    )
[PY7]   18:                ;                    ( Berei
           ch                    )
```

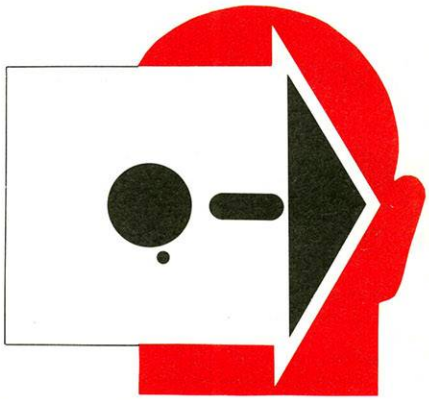
```
[000]   19:
[G7G]   20: : vectlo ( var --- loaddr )          ( Adres
          se von LO Wort des       )
[HAA]   21:                @                     ( Vekto
          rs auf den die           )
[A4B]   22:                ;                     ( Varia
          ble zeigt auf TOS legen )
[000]   23:
[A6F]   24: : vecthi ( var --- hiaddr )          ( dto.
          Adresse HI Wort          )
[2E0]   25:                @ 2+
[V10]   26:                ;
[000]   27:
[YSM]   28: : getvect ( intnum var --- )         ( Inter
          ruptvektor holen und sichern )
[2M4]   29:                [ $53 ]       ( push bx )
[EL4]   30:                [ $06 ]       ( push es )
[421]   31:                swap
[N32]   32:                $00ff and
[2N1]   33:                $3500 or
[EY3]   34:                [ $cd $21 ]   ( int 21  )
[2M4]   35:                [ $53 ]       ( push bx )
[EL4]   36:                [ $06 ]       ( push es )
[T84]   37:                [ $58 ]       ( pop ax  )
[K21]   38:                over
[RC2]   39:                vecthi !
[YMO]   40:                dup
[T84]   41:                [ $58 ]       ( pop ax  )
[421]   42:                swap
[FE2]   43:                vectlo !
[R74]   44:                [ $07 ]       ( pop es  )
[FF4]   45:                [ $5b ]       ( pop bx  )
[V10]   46:                ;
[000]   47:
[000]   48:
[TZG]   49: : setvect ( intnum var --- )  ( Interruptve
          ktor setzen )
[7M4]   50:                [ $52 ]       ( push dx )
[SU4]   51:                [ $1e ]       ( push ds )
[T83]   52:                dup vecthi @
[290]   53:                >r
[FE2]   54:                vectlo @
[290]   55:                >r
[N32]   56:                $00ff and
[ON1]   57:                $2500 or
[KF4]   58:                [ $5a ]       ( pop dx  )
[2F4]   59:                [ $1f ]       ( pop ds  )
[EY3]   60:                [ $cd $21 ]   ( int 21  )
[2F4]   61:                [ $1f ]       ( pop ds  )
[KF4]   62:                [ $5a ]       ( pop dx  )
[N21]   63:                drop
[V10]   64:                ;
[000]   65:
[B5B]   66: macro ds@      ( DS Registerinhalt auf TOS )
[YMO]   67:                dup
[YY4]   68:                [ $8c $d8 ]   ( mov ax,ds )
[V10]   69:                ;
[000]   70:
[35B]   71: macro cs@      ( CS Registerinhalt auf TOS )
[YMO]   72:                dup
[RY4]   73:                [ $8c $c8 ]   ( mov ax,cs )
[V10]   74:                ;
[000]   75:
[D3D]   76: ( beginint < --- >            Definition im
          Compiler )
[NWC]   77: ( endint   < --- >            Definition im
          Compiler )
[HSE]   78: ( exitto   < vectvar  --- >   Definition im
          Compiler )
[EME]   79: ( callto   < vectvar  --- >   Definition im
          Compiler )
[6FD]   80: ( find     < --- addr>        Definition im
          Compiler )
[V5D]   81: ( int      < intno --- >      Definition im
          Compiler )
[000]   82:
[L03]   83: : keep ( --- )
[LHO]   84:                s0 @
[TDO]   85:                16 /
[740]   86:                1+
[290]   87:                >r
[DPO]   88:                $3100
[NH4]   89:                [ $5a ]     ( pop dx )
[P24]   90:                [ $cd $21 ] ( int 21 )
[V10]   91:                ;
[000]   92:
[5S2]   93: (save.voc)
[000]   94:
[662]   95: : main ;
```

Listing 1. Das Vocabulary »isys« zur Interrupt-Programmierung

```
[WWC]    0: ( Beispielprogramm für ISYS-Vocabulary )
[34C]    1: ( für speicherresidente Programme      )
[KU9]    2: ( Autor: Erwin Schomburg               )
[000]    3:
[MF4]    4: include forth
[804]    5: include isys
[000]    6:
[4HC]    7: ( es folgt ein kleines Beispiel, wie man  )
[MWB]    8: ( dem Drucker ein X für ein U vormacht... )
[000]    9:
[YSC]   10: constant CHA1          0       ( Druckerkan
          al: LPT1          )
[99G]   11: constant PRINTER     $17       ( Interrupt
          Parallelport      )
[8FG]   12: constant PARPRT        0       ( Subfunkt.
          'Zeichen an Drucker')
[000]   13:
[PL6]   14: variable oldint        0
[C26]   15: variable pproc         0
[000]   16:
[SE3]   17: : printproc
[WXC]   18: beginint                       ( Register s
          ichern            )
[000]   19:
[6PD]   20:  dx @ CHA1 = if                ( wird LPT1
          angesprochen?     )
[W62]   21:    ax @ dup
[VGF]   22:    $ff00 and PARPRT = if       ( Kommando '
          Zeichen drucken?  )
[000]   23:
[MTC]   24:      $00ff and 'U' = if        ( ist Zeiche
          n = 'U'?  dann    )
[66F]   25:       ax @ $ff00 and 'X' + ax ! ( ein 'X' fü
          r ein 'U' vormachen )
[FG1]   26:     endif
[FG1]   27:   endif
[FG1]   28:  endif
[A5G]   29:  oldint exitto                 ( Originalen
           Druckerinterrupt   )
[C66]   30:                                ( anspringen
                             )
[V10]   31: ;
[000]   32:
[000]   33:
[6Z1]   34: : main
[UMO]   35:  hex
[C1C]   36:  ." Drucker Interrupt Umleitung von " cr
[34F]   37:  PRINTER oldint dup makevect  ( Variable al
          s Vektor-         )
[1N7]   38:                               ( Pointer vor
          bereiten,         )
[HLC]   39:  getvect                      ( Druckerinte
          rrupt-            )
[216]   40:                               ( Vektor hole
          n                 )
[U95]   41:                               ( und sichern
                            )
[000]   42:
[AT2]   43:  oldint dup
[XK4]   44:  vecthi @ 4 u.r ." :"
[XO4]   45:  vectlo @ 4 u.r
[YP4]   46:  pproc makevect
[SMC]   47:  cs@ pproc vecthi !  ( eigenes Codesegment
              )
[K38]   48:                      ( holen und als HI Wor
          t  )
[ZP6]   49:                      ( des Pointers setzen
              )
[000]   50:
[SRC]   51:  find printproc      ( Wortadresse suchen
              )
[D1B]   52:  pproc vectlo !      ( und als LO Wort des
              )
[CN6]   53:                      ( Pointers setzen
              )
[000]   54:
[L5E]   55:  PRINTER pproc setvect  ( Interruptvektor a
          uf      )
[0F8]   56:                          ( eigene Prozedur s
          etzen )
[R62]   57:  ."  nach "
[XH2]   58:  pproc dup
[XK4]   59:  vecthi @ 4 u.r ." :"
[FB4]   60:  vectlo @ 4 u.r cr
[G05]   61:  ." benötigt (Hex) "
[UE4]   62:  s0 @ 4 u.r ."  Bytes" cr
[VH8]   63:  keep      ( Programm resident      )
[7E4]   64:            ( machen                 )
[V10]   65: ;
```

Listing 2. Ein Beispiel für die Anwendung des Isys-Vocabularies

Sesam öffne dich!



BCi DisAsm erschließt jedes Programm!

Wandelt dies

```
D>DEBUG HALLO.COM
-D100 160
1CB0:0100  EB 43 90 48 61 6C 6C 6F-20 46 72 65 75 6E 64 65   .C.Hallo Freunde
1CB0:0110  21 24 0D 0A 56 69 65 6C-20 45 72 66 6F 6C 67 20   !$..Viel Erfolg
1CB0:0120  6D 69 74 20 42 43 49 20-44 69 73 41 73 6D 2F 69   mit BCI DisAsm/i
1CB0:0130  38 36 2E 24 0D 0A 47 72-81 E1 65 20 76 6F 6E 20   86.$..Gr..e von
1CB0:0140  42 43 49 2E 24 BA 03 01-B4 09 CD 21 BA 12 01 B4   BCI.$......!....
1CB0:0150  09 CD 21 BA 34 01 B4 09-CD 21 CD 20 96 B9 B9 10   ..!.4....!. ....
1CB0:0160  00
-
```

in diese Form um!

und kann nach Modifizierung oder Änderung durch einen Editor mit gängigen Assemblern oder Debuggern wieder assembliert werden.

```
BCi DisAsm/i86 CROSSREFERENZ  Version 1.0 - HALLO.XRF                    Seite 1

00103h                  :    145
00112h                  :    14C
00134h                  :    153
BIOS-TERM_norm:20h      :    15A
DOS1-Dsp_Str            :    14A   151   158
H00145                  :    100

ANZAHL BEGRIFFE AUF DER CROSSREFERENZLISTE: 6 - 4


;NAME:              HALLO.COM
;DATEILAENGE:       0005Ch - 92d
;START (CS:IP):     00100h
;CODE ENDE:         0015Ch
;CODE ANFANG:       00100h
;DATUM:             Tue Dec 01 07:14:28 1987

CODE    SEGMENT BYTE PUBLIC 'CODE'
ASSUME  CS:CODE,DS:CODE,ES:NOTHING,SS:NOTHING

P00100  PROC
        ORG     0100h

H00100: JMP     Short H00145                    ;00100 EB43          _C
;-------------------------------------------------
        NOP                                     ;00102 90            _
;-------------------------------------------------
        DB      "Hallo Freunde!$"               ;00103 48616C6C6F2046
        DB      0Dh                             ;00112
        DB      0Ah                             ;00113
        DB      "Viel Erfolg mit BCI DisAsm/i86.$" ;00114 5669656C204572
        DB      0Dh                             ;00134
        DB      0Ah                             ;00135
        DB      "Gr"                            ;00136 4772
        DB      81h                             ;00138
        DB      0E1h                            ;00139
        DB      "e von BCI.$"                   ;0013A 6520766F6E2042
;-------------------------------------------------
H00145: MOV     DX,0103h                        ;00145 BA0301        ___
        MOV     AH,09h                          ;00148 B409          __
        INT     21h             ;1-Dsp_Str      ;0014A CD21          _!
        MOV     DX,0112h                        ;0014C BA1201        ___
        MOV     AH,09h                          ;0014F B409          __
        INT     21h             ;1-Dsp_Str      ;00151 CD21          _!
        MOV     DX,0134h                        ;00153 BA3401        _4_
        MOV     AH,09h                          ;00156 B409          __
        INT     21h             ;1-Dsp_Str      ;00158 CD21          _!
        INT     20h             ;B-TERM_norm:20h ;0015A CD20         _
;-------------------------------------------------

P00100  ENDP

CODE    ENDS
        END     H00100

;-------------------------------------------------------------------------------
```

BCi DisAsm
Es gibt kaum einen, der ihm das Wasser reichen kann!

Auch Experten kennen nicht immer die Lösung, aber sie wissen, wo sie suchen müssen.
BCi DisAsm hilft JEDEM, die verwirrenden und frustrierenden Software-Puzzle, welche sich beim Umgang mit dem Computer ergeben, zu lösen. Und zwar schneller als mit jedem anderen Produkt, das Sie erwerben können, egal welche Preisklasse Sie betrachten.
Mit dem BCi DisAsm erwerben Sie Know-How, das Sie sonst Jahre an Erfahrung kosten würde. Versuchen Sie nicht, das Rad neu zu erfinden!

BCi DisAsm – Der intelligente Disassembler
BCi DisAsm ist ein intelligenter, selbsttätig dokumentierender MS-DOS Disassembler. Die ausgefeilten Routinen im BCi DisAsm analysieren in kürzester Zeit selbst komplexen und umfangreichen Programmcode und zeigen dabei potentielle Problempunkte auf.

BCi DisAsm – Das Lernmittel
Sie brauchen kein Spezialist in Assembler-Programmierung zu sein. BCi DisAsm produziert aus jedem beliebigen Programm ihres IBM-kompatiblen Computers ein gut dokumentiertes, leicht verständliches Assemblerlisting.

BCi DisAsm – Einfach in der Handhabung
BCi DisAsm kann sowohl über Parameter von der DOS-Kommandozeile aus als auch interaktiv über Menues gesteuert werden. Die Menuestruktur ist Lotus-kompatibel, selbstverständlich in Deutsch. Es besteht keine Notwendigkeit, neue Befehle zu lernen oder ständig im Handbuch nachzuschlagen. BCi DisAsm ist unmittelbar anwendbar!

Minimale Systemanforderungen
256 kB RAM, 8088/8086/80186/80286 oder 80386 CPU,
MS-DOS oder PC-DOS Version 2.0 oder größer

BCi DisAsm wird ihnen bei den folgenden Schritten helfen:

1. Lernen Sie Assembler-Programmierung, wenn Sie möchten.
2. Entdecken Sie, warum Programm XYZ auf ihrem Rechner nicht läuft und tun Sie etwas dagegen.
3. Ändern oder entfernen Sie Programmteile.
4. Sichern Sie sich gegen destruktive Programme ab. Finden Sie einen "Virus" und entfernen Sie ihn.
5. Integrieren Sie Code aus compilierten Programmen in eigene Anwendungen.
6. Erhöhen Sie die Kompatibilität Ihrer Software.
7. Passen Sie Software an verschiedene MS-DOS-Versionen an.

© 1987 BCi, West Germany und USA
Lotus ist eingetragenes Warenzeichen der Lotus Corp.
MS-DOS ist eingetragenes Warenzeichen der Microsoft Corp.

BCi DisAsm kostet incl. einem sehr ausführlichen deutschen Handbuch (mehr als 100 Seiten) nur **(Best.-Nr. 227)**

199,– DM*

* unverbindliche Preisempfehlung

Bestellen Sie noch heute mittels der entsprechenden Postkarte bei:

DMV Software

Postfach 250, 3440 Eschwege, Tel. 05651/8702

Händleranfragen willkommen

Impressum

Herausgeber
Christian Widuch

Chefredaktion
Andreas Hagedorn (hg), Volker Everts (ev)

Redaktion
Martin Aschoff (ma), Christian Spitzner (sp)

Schlußredaktion
Eva-Maria Hierlmeier, Andrea Scheding

Koordination
Stefan Ritter

Produktionsleitung
Gerd Köberich

Satz
Claudia Küllmer, Silvia Führer, Martina Siebert
Gabriela Joseph

Lektorat
Elvira Domonkos, Susanne Mias

Gestaltung
Yvonne Hendricks, Manuela Eska, Mohamed Hawa

Reprografie
Margarete Schenk, Helmut Skoupy

Illustration
Heinrich Stiller

Fotografie
Christian Heckmann

Anzeigenleitung
Wolfgang Schnell
Es gilt die Anzeigenpreisliste Nr. 1 vom 22.09. 1987

Anzeigenverkauf
Wolfgang Brill

Anschrift Redaktion:
Redaktion DOS International,
Elfenstr. 40, 8000 München 83
Tel. (089) 60 20 66

Anschrift Verlag:
DMV-Daten & Medien Verlagsges. mbH
Postfach 250, Fuldaer Straße 6
3440 Eschwege
Telefon: (0 56 51) 87 02
Telex 993 210 dmv d
Telefax 0 56 51 / 2 07 18

Vertrieb
Verlagsunion
Friedrich-Bergius-Straße 20
6200 Wiesbaden

Druck
Druckerei Jungfer, 3420 Herzberg

Bezugspreise
DOS EXTRA erscheint viermal jährlich,
Einzelpreis 18,- DM, 145 Ös, 18,- sfr.

Bankverbindungen:
Postscheck Frankfurt/M: Kto.-Nr.: 23043-608
Raiffeisenbank Eschwege:
BLZ: 522 603 85, Kto.-Nr.: 245 7008

Manuskripte, Programme:
bitte an die Redaktion DOS International,
Elfenstr. 40, 8000 München 83 senden.

Für unverlangt eingesandte Manuskripte und Datenträger sowie Fotos übernimmt der Verlag und die Redaktion keine Haftung. Die Zustimmung zum Abdruck wird vorausgesetzt.
Das Urheberrecht für veröffentlichte Manuskripte liegt ausschließlich beim Verlag. Nachdruck sowie Vervielfältigung oder sonstige Verwertung von Texten, nur mit schriftlicher Genehmigung des Verlages.
Namentlich gekennzeichnete Fremdbeiträge geben nicht in jedem Fall die Meinung der Redaktion wieder.

Wissen Sie mehr von PCs als andere Leute?

Dann schreiben Sie uns. Die Redaktion DOS International sucht aktive Mitarbeiter, die über interessante Gebiete aus der »Computerei« berichten wollen. MS-DOS, DOS Plus, GEM, Drucker, Basteleien – es gibt nichts, worüber nicht andere Leser unserer Zeitschrift auch etwas lesen wollen.

Ihren Einsatz lassen wir uns natürlich etwas kosten. Ihr Manuskript wird selbstverständlich honoriert. Bilder, Zeichnungen, Tabellen und Listings werden gesondert berechnet.

Doch bevor Sie sich hinsetzen und das Schreiben anfangen:
Rufen Sie uns an.

Sagen Sie uns, was Sie schreiben wollen, und wir klären ob – und wann – der Artikel in unser Konzept paßt.

Der Artikel muß mit einem Textsystem erfaßt werden. Wir brauchen von Ihnen eine Diskette im MS-DOS-Format, auf der der Text als reines ASCII-Feld steht.

Schreiben Sie uns auch, wenn Sie nur Vorschläge für Themen haben.

DOS International ist nämlich eine Zeitschrift von aktiven PC-Besitzern für aktive PC-Benutzer.

Besonders das DOS-Forum lebt von Ihrer Mitarbeit. Auch die Redaktion kann nicht alle Antworten wissen. Fragen an uns, die nicht sofort beantwortet werden können, werden im DOS-Forum veröffentlicht. Wissen Sie eine Antwort, so wird diese veröffentlicht.

Sie sehen: DOS International ist die PC-Zeitschrift, in der das Wissen der PC-Benutzer untereinander ausgetauscht wird.

Redaktion
DOS International
Elfenstr. 40
8000 München 83
Tel.: (089) 60 20 66

DOS-EXTRA DATABOX

— Die Software zur DOS-EXTRA Nr. 2 —

auf 4 Disketten im Format 5 1/4“ oder 2 Disketten im Format 3 1/2“

Sagenhaft günstig, extra viel gute Software für’s Geld!

DATABOX:

- mehr als der übliche Softwareservice
- bringt alle Programme der DOS EXTRA auf Datenträger

Für alle PCs unter MS-DOS als 5 1/4“ oder 3 1/2“ Diskette

Die Databox bietet unter anderem:

- **DOS-Commander – eine Benutzeroberfläche für MS-DOS**
- **SCopy – ein professionelles Kopierprogramm**
- **SForth 3.1: Ein Compiler im Selbstbau**
- **SCAD – 3D-Grafik selbst gemacht**
- **Hurrican – ein Actionspiel der Superlative**
- **Supermind – ein Strategiespiel**
- **Utilities: Routinen für die tägliche Arbeit**
- **SCheck: Checksummer für die Eingabe**

und des weiteren die lauffähigen Dateien aller im Heft abgedruckten Listings

EXTRA-PREIS: 4 Disk 51/4“ 45,– DM
2 Disk 3 1/2“ 45,– DM

DMV-Verlag · Postfach 250 · 3440 Eschwege

— Bitte Bestellkarte benutzen —

MCI

PERSONAL COMPUTER

Bitte neueste Prospekte anfordern!

Auf alle Geräte 12 Monate Garantie. Preise gültig ab 1.12.87. Lieferbedingungen auf Anfrage. MCI MICRO COMPUTER INSTRUMENTS GMBH eingetragen AG Bergisch Gladbach HRB 2575 · Herstellung und Vertrieb von Microcomputern.

MCI

Bensberger Straße 252 · 5060 Bergisch Gladbach 2
Tel. (02202) 1080
Fax: (02202) 31009 · Telex: 8873518